VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce vzorů v byznys procesech

Pattern detection in business processes

2014

Zbyněk Vrána

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Zbyněk Vrána**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Detekce vzorů v byznys procesech
Pattern Detection in Business Processes

Zásady pro vypracování:

Cílem práce je provedení průzkumu existujících přístupů v oblasti automatické detekce vzorů v byznys procesech, návrh a implementace vybrané nebo vlastní metody a aplikačního prostředí pro experimenty.

1. Průzkum a popis existujících přístupů.
2. Návrh a implementace vybrané nebo vlastní metody.
3. Návrh a implementace počítačové aplikace pro provádění experimentů.
4. Návrh, realizace a hodnocení experimentů.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Miloš Kudělka, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2014 .....

Děkuji panu Mgr. Miloši Kudělkovi, Ph.D. za cenné rady při konzultacích a podporu při překonávání počáteční paniky.

Abstrakt

Tato práce popisuje možnosti vyhledání procesních vzorů v modelech byznys procesů vytvořených s pomocí standardu BPMN. Jedním z hlavních pilířů této práce je navázání na výzkum profesora Vin van der Aalsta z univerzity Eidhoven v oblasti procesních vzorů a využití těchto podkladů pro stanovení, zda jsou ve zkoumaném modelu použity procesní vzory a v jakém rozsahu. V úvodních kapitolách jsou představeny standardy, použité při modelování byznys procesu a jednotlivé skupiny vzorů se stručným popisem. V dalších kapitolách následuje popis vybraného algoritmu pro detekci vzorů a aplikace, která funkci algoritmu ověří v praktickém použití.

Klíčová slova: Vzor, proces, detekce, graf, matice, isomorfismus, BPMN, SQL

Abstract

This thesis describes the process of finding workflow patterns in models of business processes created using the BPMN standard. One of the main pillars of this work is to built on the research in the field of workflow patterns published by professor Vin van der Aalst from Eindhoven University and using his work to determine whether the model is using process patterns and to what extent. The introductory chapters describe standards used to model business process and individual groups of patterns with a brief description. In subsequent chapters follows a description of selected patterns detection algorithm and an application that verifies the algorithm in practical use.

Keywords: Pattern, workflow, detection, graph, matrix, isomorphism, BPMN, SQL

Seznam použitých zkratek a symbolů

ARIS	–	Nástroj pro analýzu podnikových procesů, které jsou modelovány pomocí různých pohledů
BPMN	–	Soubor principů a pravidel, který slouží pro grafické znázorňování podnikových procesů pomocí procesních diagramů
EA	–	Aplikace Enterprise Architect, určená pro modelování UML a BPMN diagramů
IDEF	–	Modelovací jazyk pro vizualizaci a širokou specifikaci systémů
UML	–	Grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů
SQL	–	Standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích
WPI	–	Iniciativa, jejímž hlavním zaměřením je výzkum postupů v systémech pro byznys modelování

Obsah

1 Úvod **2**
1.1 Historie Workflow Patterns Initiative a procesních vzorů 2

2 Používané metody pro modelování byznys procesů **3**
2.1 Modelovací standard BPMN 3

3 Procesní vzory v byznys modelování **4**
3.1 Představení procesních vzorů 4
3.2 Řídící vzory 4
3.3 Vzory pro práci se zdroji 9
3.4 Vzory pro práci s daty 10
3.5 Vzory pro zpracování výjimek 10
3.6 Prezentační vzory 12
3.7 Použité prvky pro modelování 12

4 Vzory v modelovacích systémech **13**

5 Detekce vzorů v modelu **15**
5.1 Metody ověření izomorfismu grafu a podgrafu 16
5.2 Reprezentace grafu pomocí matice incidence 18
5.3 Postup hledání podmatice vzoru v matici modelu 18
5.4 Omezující vlastnosti algoritmu 21
5.5 Výjimky pro nahrazení vzorů základním objektem 22

6 Testovací aplikace **23**
6.1 Specifikace požadavků a scénáře případů užití 23
6.2 Datové rozhraní aplikace Enterprise Architect 25
6.3 Kvalitativní požadavky 27
6.4 Ukázky výstupů zpracování testovaných dat 27

7 Závěr **28**

8 Reference **29**

Přílohy **29**

A Testy detekce vzorů v rozsáhlých modelech **30**
A.1 První ověření modelu z praxe 30
A.2 Test modelu kontroly dokladů 33
A.3 Test modelu hospitalizace pacienta 36

B Uživatelská příručka k aplikaci **38**
B.1 Export dat z aplikace Enterprise Architect a uživatelská konfigurace . . . 39

1 Úvod

Pro každou fungující organizaci je velmi důležité zvládnutí řízení vnitřních procesů, jehož obtížnost a význam exponenciálně roste s rozsahem strukury firmy a množstvím vnitřních firemních procesů. K tomuto účelu se pak používá celá řada technik, které umožňují na abstraktní úrovni modelovat tyto firemní procesy tak, aby se reprezentovaný model mohl využít pro další zpracování. Může to být například vylepšování vybraných procesů, zaučování nových pracovníků, identifikace kompetencí a zodpovědnosti pracovníků, zavedení automatizace řízení a celkově zefektivnění zvládnutí procesního řízení. Analytik, který je postaven před úkol specifikovat byznys procesy v nějaké společnosti, když po splnění tohoto úkolu porovná své výsledky s jiným analytikem, který ve stejné společnosti prováděl stejnou činnost, zjistí, že většina činností v porovnávaných modelech je zapsaná shodně, nebo velmi podobně. Reflektuje to skutečnost, že model respektuje stávající podnikové procesy, které v organizaci neustále probíhají, bez ohledu na to, zda jsou někde popsané, anebo ne. Při dalším zkoumání a porovnávání prací obou analytiků je velmi pravděpodobné, že některý z popisů procesů bude mít jeden z analytiků přehledněji a jednodušeji zapsaný než analytik druhý. Zkušenější analytik, který v oboru působí déle a již se dříve setkal s modelováním podobných procesů má stanoveny optimalizované postupy pro řešení současných problémů. Tyto zkušenosti také umožní analytikovi vyhnout se řešením vyloženě špatným, které můžeme charakterizovat jako antivzory.

Specifikace optimalizovaných postupů (vzorů) při modelování byznys procesů a jejich praktického využití je hlavním cílem této práce. Při její realizaci vycházím z práce profesora Wil van der Aalsta, jehož práci v oblasti procesních vzorů pro modelování byznys procesů považuji za jednu z nejkomplexnějších. [1][cit. 3.3.2014]

1.1 Historie Workflow Patterns Initiative a procesních vzorů

První krůčky k procesním vzorům se datují k polovině devadesátých let, kdy profesor Wil van der Aalst v rámci projektu Sagitta s údivem pozoroval, s jakými nepřesnostmi se zadavatelé potýkají při popisu požadovaných funkcí systému. Na základě této zkušenosti, v roce 2000 na konferenci CoopIS v Edingburgu představil pojednání, jehož základem bylo představení první základní skupiny procesních vzorů. To byl důležitý impuls pro založení stránek www.workflowpatterns.com a sdružení Workflow Patterns Initiative (WPI), jehož cílem je systematické zkoumání komerčních i nekomerčních modelovacích systémů a standardů a evidence používaných procesních vzorů. V této době se pozornost upínala především na řídící procesní vzory, což se změnilo v roce 2004 příchodem Nicka Russella do WPI. Russell představil další skupiny vzorů, například vzory pro práci s daty, se zdroji, ale také specifické vzory pro ošetření výjimek během vykonávání procesů. Iniciativa WPI je v současnosti již stabilní základnou v rámci procesních vzorů, z počátečního počtu 20 procesních vzorů je k aktuálnímu roku 2014 v evidenci WPI 147 podrobně popsaných vzorů.

2 Používané metody pro modelování byznys procesů

Existuje velmi široká škála možností, jak byznys procesy popsat. Nejjednodušší je neformální specifikace, kterou může být slovní popis doplněný obrázky a která je charakteristická pro první fáze analýzy procesů. Tyto specifikace jsou pak většinou přepracovány v semiformální specifikaci, která již obsahuje přesně popsaný daný systém s jednoznačnou syntaxí způsobu popisu. Mezi nejznámější semiformální modelovací standardy se řadí BMPN, IDEF, ARIS nebo UML.[4] Od roku 2003 pak přímo pro účely výzkumu v oblasti procesních vzorů vyvíjí WPI vlastní otevřený modelovací systém YAWL.

Vyšší formou popisu jsou formální metody, které mají nejen jednoznačnou syntaxi a sémantiku, ale samotná specifikace je také ověřena, zda obsahuje požadované vlastnosti. Jelikož jejich použití vyžaduje velké úsilí a investici velkého množství času (specifikace musí být napsána bezchybně a pokrývat všechny možnosti), jsou použity jen v takových případech, kde je bezchybný chod procesů nutně vyžadován. Nejznámější formální metody jsou konečné automaty nebo Petriho sítě.

V této práci bude pro modelování vzorů i testovacích modelů použit standard BPMN 2.0 [2] s modelováním v aplikaci Enterprise Architect.

2.1 Modelovací standard BPMN

Hlavním cílem BPMN je poskytnout notaci pro pochopení a znázornění podnikových procesů v přehledném grafickém zápisu standardizovaným způsobem. K modelování procesů se používají vzájemně propojené grafické prvky - objekty, kde každý z nich má pevně stanovený význam. Tyto objekty se dělí do čtyř kategorií:

- **Tokové objekty** jsou hlavní prvky, které přímo definují chování procesů. Události představují děj, který nastane během vykonávání procesu nebo na jeho začátku a konci a přímo tak ovlivní tok procesu nebo načasování aktivit. Aktivita je obecná forma pro vyjádření činnosti, která se vykonává uvnitř procesu. Brány jsou určeny pro řízení rozdělení nebo spojení sekvence toků v procesu.

- **Spojovací objekty** spojují vzájemně ostatní objekty. Sekvenční tok a tok zpráv má vždy určen směr, který určuje posloupnost vykonávávání spojených objektů. Výjimkou je asociace bez určeného směru, která je ale většinou spojena s jiným spojovacím objektem, který již ukazuje směr.

- **Plavecké dráhy** slouží k rozlišení odpovědnosti jednotlivých aktérů, nebo organizování a kategorizování aktivit. Komunikace mezi jednotlivými dráhami pak probíhá pomocí spojovacích objektů.

- **Artefakty** umožňují přidat do modelu další informace, jako jsou poznámky, datové objekty nebo rozlišení skupin aktivit bez ovlivnění toku procesu.

3 Procesní vzory v byznys modelování

Máme možnost kreslit modely procesů nejen na papír, ale je k dispozici řada různě vyspělých softwarových systémů, které modelování procesu výrazně usnadní. K tomuto účelu využívají různé normy (notace), u kterých ale musí respektovat, že výsledné modely musí být interpretovány vždy stejně, bez ohledu na softwarovou platformu, na které byly modelovány. Většinou se jedná o semiformální standardy, kde pak některé systémy umožňují převod na Petriho sítě a kontrolu validnosti zadaného modelu.

Všechny tyto systémy podporují procesní vzory, ale obecně jen určitou část z celkového množství možných vzorů. Je to dáno tím, že některé vzory jsou natolik abstraktního a specifického zaměření, že není možné jejich zobecnění, anebo naopak, systém má natolik specifické použití, že řadu vzorů ve své podstatě modelovat nemůže. Vždy jsou ale téměř bez výjimek podporovány všechny základní (atomické) řídící vzory, protože bez nich by nebylo vůbec možné modelovat průběh a návaznosti procesů.

3.1 Představení procesních vzorů

Procesní vzor v byznys modelování představuje řazení sekvence úloh a výběru rozhodnutí v průběhu procesu podle předem definované šablony - vzoru. Vzor jako takový je definován jako abstraktní a generalizované vyjádření, takže například pro modelování procesů které provádí sekretářka vyřizující poštu a procesů výroby v nějaké montážní hale můžeme použít jeden a tentýž vzor. Řada základních vzorů je již pevně definována v některých modelovacích normách, například rozhodovací prvky pro rozdělení toku. Způsob vyjádření vzoru není pevně vázán na určitou normu a lze použít většinu specifikací od neformálních až po formální.

Vzory jsou rozčleněny do několika skupin, podle toho, s jakými zdroji nebo úlohami pracují. Rozdělení také plyne ze skutečnosti, v jaké fázi analýzy se s daným modelem pracuje. Vzory pro práci se zdroji lze použít ve chvíli, kdy jsou definováni aktéři, kteří budou v rámci rolí v modelu specifikováni. Stejné pravidlo se pak uplatní pro vzory, které pracují s daty. Prezentační vzory je možné uplatnit až ve chvíli, kdy je již k dispozici ucelený model.

Následující přehled vzorů a jejich skupin poskytuje podrobnější pohled jen u těch skupin a vzorů, které vyhovují možnosti použití algoritmu pro detekci vzorů, který je předmětem této práce. Ostatní skupiny a vzory jsou popsány jen v obecné rovině.

3.2 Řídící vzory

Úkolem řídících vzorů je ovlivňování průběhu zpracování byznys procesů. Obsahuje možnosti pro rozdělování toku procesu, jeho opětovné spojení, zastavení zpracování procesu na určitou dobu, nebo trvale. Podrobný popis řídících vzorů podle AALSTA [1][cit.16.3.2014] je dostupný z WWW,

$< http://www.workflowpatterns.com/patterns/control/ >$.

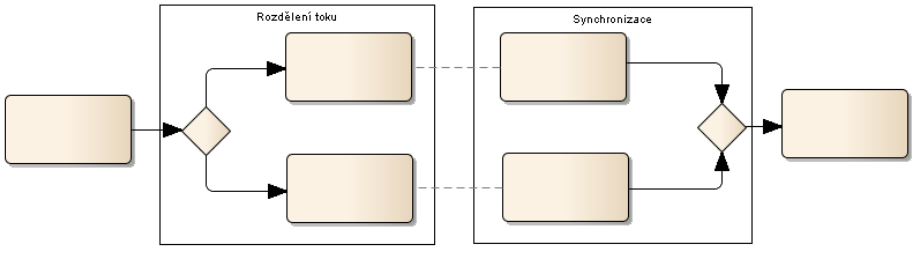
Základním stavebním kamenem každého modelu je vzor **Sequence**, kde jde o sérii úloh, které se vykonávají přímo jedna za druhou. Nutné je dodržet podmínku, že v jednom okamžiku může být zpracovávána jen jedna instance, nebo vlákno vzoru. Dalšími základními vzory, které jsou obsaženy ve všech notacích pro modelování byznys procesů je rozdělení toku procesu. Rozdělení toku se provede na základě určité podmínky, ale je také řada vzorů, které takto rozdělené toky později opět spojí. Jestliže je vyžadováno, aby všechny rozdělené toky nejprve doputovaly do spojovacího objektu a teprve pak je povoleno pokračovat jednomu toku dál, jedná se o synchronizované spojení. U nesynchronizovaného spojení projde tok spojovacím objektem bez čekání na toky v ostatních větvích. Následující tabulky poskytují přehled nad těmito vzory, které jsou rozděleny na skupiny podle rozhodovacích podmínek v rozdělení nebo spojení toků:

Podmínka pro rozdělení toku	**Název vzoru pro rozdělení toku**
Tok se rozdělí na dvě a více větví, které se vykonávají souběžně	Pararell Split
Vybere se jen jedna větev, kde se tok bude vykonávat	Exclusive Choice
Větve pro provádění toku se určí podle rozhodovacích podmínek	Multi Choice

Tabulka 1: Vzory pro rozdělení toku

Vzor rozdělující tok	**Vzor pro synchronizované spojení**	**Vzor pro nesynchronizované spojení**
Pararell Split	Synchronization	Partial Join
Exclusive Choice	Synchronizing Merge	Simple Merge
Multi Choice	Structured Synchronizing Merge	Multi-Merge

Tabulka 2: Rozdělení vzorů pro spojení toků



Obrázek 1: Ukázka rozdělení toku a následná synchronizace

Vytvoření více úloh v rámci jedné instance procesu zajišťuje vzor **Thread Merge**. Opakem tohoto vzoru je **Thread Split**, který sloučí v rámci jedné instance procesu zvolený počet přicházejících úloh do jedné odchozí úlohy. Instance procesů mohou vytvářet nové úlohy, ale také můžou vytvářet nové instance procesů. Touto problematikou se zabývá skupina vzorů **Multiple Instance**.

Jestliže je potřeba řídit spojení více pararelně prováděných procesů, využívá se k tomuto účelu **Discriminator**. Ten umožní některé procesy v závislosti na stavu jiných procesů pozastavit, nebo úplně zrušit. Rozdíl vzorů **Partial Join** proti vzorům používajících Discriminator spočívá ve stanoveném minimálním počtu procesů, čekajících na synchronizaci ve spojovacím prvku. Zde platí vztah $2 \leq n < m$, pro n = počet procesů čekajících na synchronizaci, m = počet větví, vstupujících do spojovacího prvku.

Pravidla pro provedení procesů	**Název vzoru**
Dokud vybrané procesy neprojdou spojovacím prvkem, jsou ostatní procesy před spojením pozastaveny	Structured Discriminator/Partial Join
Varianta vzoru Structured Discriminator, kdy je ale umožněno pozastavit některé procesy kdekoliv v modelu. Tento vzor je vhodný pro prostředí, kde se souběžně zpracovává více vláken v jedné instanci procesu	Blocking Discriminator/Partial Join
Při synchronizaci spojení více procesů se nemusí čekat na všechny zdržené probíhající procesy, které spojovacím prvkem prochází. Procesy, které se spojení neúčastní jsou pak ve svém provádění zrušeny	Cancelling Discriminator/Partial Join

Tabulka 3: Vzory pro řízení spojení procesů

3.2.1 Stavové vzory

U stavových vzorů není rozhodnutí vždy pevně dané, kterou větev z brány si proces má vybrat, ale závisí na okamžitém stavu jiných prvků v prostředí procesu. Vzor **Deferred Choice** je variantou Exclusive Choise, s doplněním podmínky, kdy výběr větve během rozdělení toku rozhodne stav jiného, předem definovaného elementu. Jinou variantou vzoru Sequence je vzor **Interleaved Parallel Routing**. který umožňuje změnit pořadí vykonávání úloh, v závislosti na stavech obsažených prvků v řadě. Jestliže je potřeba zajistit, aby se určitá úloha zpřístupnila jen v okamžiku, kdy je celá instance procesu v určitém stavu (je aktivována vybraná část procesu), pro tento účel je určen vzor **Milestone**. V rámci vzoru **Critical Section** je možné vymezit několik částí procesu, které jsou považovány za "kritické sekce"a platí, že v rámci instance procesu může být vykonávána v jednom okamžiku jen jedna z kritických sekcí. Vzor **Interleaved Routing** zajistí, že se vybraná řada úloh vykoná v jakémkoliv pořadí, ale žádná z úloh se neprovede více než jednou.

3.2.2 Vzory pro správu více instancí procesů

Základní vlastností těchto procesů je možnost vytvořit řadu úloh v rámci svého objektu, jejichž počet může být na začátku procesu předem známý, nebo je určen podle vlastností okolního prostředí, jako je dostupnost prostředků, stav dat a možné stavy procesů. Tyto úlohy jsou na sobě plně závislé a běží pararelně.

Zvláštní podtřídou těchto vzorů jsou procesy vytvářející instance jiných procesů uvnitř sebe sama. Jejich vlastnosti zachycuje následující tabulka:

Pravidla pro provedení procesů	**Název vzoru**
Počet instancí úloh je předem znám, není nutné synchronizovat pořadí jejich dokončení	Multiple Instances without Synchronization
Počet instancí úloh je předem znám, je však nutné synchronizovat dokončení probíhajících úloh před vytvořením úloh nových	Multiple Instances with a priori Design-Time Knowledge
Počet instancí úloh závisí na vlivu prostředí. Úlohy jsou na sobě plně závislé, je nutná jejich synchronizace v pořadí vytváření nových instancí úloh	Multiple Instances with a priori Run-Time Knowledge
Podmínky pro provedení procesů jsou shodné se vzorem Multiple Instances with a priori Run-Time Knowledge, navíc ale lze kdykoliv v průběhu procesu přidávat nové instance úloh	Multiple Instances without a priori Design-Time Knowledge
Počet instancí úloh je na začátku provádění procesu předem znám. Jakmile je určitý počet úloh splněn, proces pokračuje dále, bez ohledu na stav ostatních rozpracovaných úloh	Static Partial Join for Multiple Instances
Počet instancí úloh je na začátku provádění procesu předem znám. Jakmile je určitý počet úloh splněn, proces pokračuje dále a ostatní rozpracované úlohy jsou zrušeny	Cancelling Partial Join for Multiple Instances
Počet instancí úloh závisí na vlivu prostředí. Jakmile je určitý počet úloh splněn, vytvoří se instance jiného typu úlohy, která se začne ihned provádět. Ostatních rozpracované úlohy se dokončí, bez dalších podmínek	Dynamic Partial Join for Multiple Instances

Tabulka 4: Vzory vytvářející instance procesů a úloh

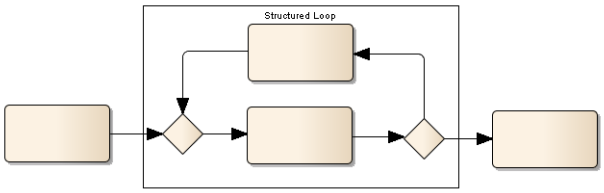
3.2.3 Vzory pro ukončení a kompletování procesů

Možnost zrušit provádění aktivní úlohy v procesu na základě určité podmínky je v možnostech vzoru **Cancel Task**. Podobnou funkci má vzor **Cancel Case**, který ale ruší celou instanci procesu. Na rozmezí mezi $Cancel\ Task$ a $Cancel\ Case$ je vzor **Cancel Region**, který umožňuje zastavit vybranou sadu úloh v rámci procesu. Je nutné zdůraznit, že tyto úlohy jen pozastaví, ale nezruší jejich instance. Vzor **Cancel Multiple Instance Task** je variantou vzoru $Multiple\ Instances$ s doplněním možnosti zrušení takto vytvořených

úloh na základě předem dané podmínky. Podobně, **Complete Multiple Instance Task** vytváří řadu nových instancí úlohy, kde po splnění určité podmínky zajistí dokončení těchto úloh a jejich vzájemnou synchronizaci (běží nezávisle na sobě) dříve, než se spustí další úloha v pořadí.

3.2.4 Iterační vzory

Nejjednodušší smyčku v rámci procesu dovoluje vzor **Arbitrary Cycles**, kde iteraci může proces libovolně opustit i jinak, než rozhodovací bránou patřící ke vzoru, nebo do ni vstoupit. Naopak, vzor **Structured Loop** nedovoluje ukončit iteraci jinak než rozhodovací podmínkou v rámci vzoru. Doplňující informace $Pre-Test$ nebo $Post-Test$ určuje, zda se test na opakování iterace bude provádět na vstupu do smyčky, anebo na jejím výstupu.



Obrázek 2: Ukázka vzoru Structured Loop

3.2.5 Vzory pro ukončení procesu

Vzor **Implicit Termination** ukončí instanci procesu ve chvíli, kdy prozkoumá možnosti dalšího pokračování procesu a žádná taková možnost není nalezena. U vzoru **Explicit Termination** je ukončena instance procesu ve chvíli, kdy je dosaženo předem určeného stavu (podle standardu BPMN je to prvek $End\ Event$). Podstatnou informací je skutečnost, že v rámci ukončení procesu pod taktovkou vzoru je bezpečně ukončena práce ještě probíhajících úloh.

3.2.6 Vzory pro správu zahájení procesů

Vzor **Transient Trigger** vytvoří instanci úlohy a zahájí její provádění ve chvíli, kdy je vyvolána interní nebo externí událost. Vyvolaná úloha musí být provedena okamžitě, pokud to nelze splnit, úloha se neprovede a je odstraněna. Podobnou funkci má vzor **Persistent Trigger**, který se ale provede vždy i za cenu toho, že úloha musí nějakou dobu čekat na své provedení.

3.3 Vzory pro práci se zdroji

Pod pojem $zdroj$ lze zařadit jak lidské zdroje (zaměstnance, uživatele atd.), tak zařízení nebo systémy. Obecně se jedná o entitu, která je schopná provést určitou úlohu, ať už samostatně, nebo v rámci nějaké skupiny. Definované zdroje pak mohou mít přiřazené přístupy k určitému výčtu částí procesu, které mohou ovlivňovat, nebo jimi mohou být ovlivňovány. Tyto přístupy pak mohou mít ještě další řadu do sebe zanořených úrovní případně tvořit složitější organizační hierarchii. V tomto kontextu lze říci, že zdroje mají přiřazené $role$ a v této souvislosti $vlastnosti$, které jim umožňují provádět nebo účastnit se vybraných úloh. Mohou byt k úlohám přiřazeny trvale nebo jen pro dílčí část vykonání úlohy. [15] Vzory pro práci se zdroji se dále člení na několik skupin:

- **Vzory pro vytváření zdrojů** - obsahují všechny vzory, u kterých se pracuje se zdroji ve chvíli, kdy je úloha vytvořena. V rámci inicializace úlohy pak vzory zajišťují kontrolní mechanismy, které k úloze přiřadí správné zdroje podle předem daných omezení a podmínek.
- **Vzory pro alokování zdrojů** - popisují v podstatě opačný proces, než vzory pro vytváření zdrojů, kdy jsou právě vytvořené úlohy přiřazovány k vybraným zdrojům.
- **Vzory pro využívání zdrojů** - v tomto případě si zdroje vybírají úlohy z nabídky v seznamu naplánovaných úloh. Zodpovědnost rozhodnutí, kdy bude zdroj přiřazen k jaké úloze je především v kompetenci zdroje.
- **Vzory pro správu využití zdrojů** - tato skupina vzorů popisuje situace, kdy z důvodu nějaké události je nutné již přidělené zdroje přerozdělit nebo změnit pravidla naplánování.
- **Vzory pro správu zahájení procesu při události** - obsahuje možnosti řešení pro situace, kdy úloha má již přiděleny zdroje, ale z důvodu nepředpokládané události je nutné tyto zdroje znovu přerozdělit, nebo přidělit nové.
- **Vzory pro správu viditelnosti zdrojů** - úloha může využít zdroje, jen pokud má k této volbě přiřazené zvláštní oprávnění.
- **Vzory pro správu zahájení procesu po přiřazení zdrojů** - zahrnuje zahájení provedení úloh za podmínek, kdy jsou dostupné nebo k úloze přímo přiřazeny určité zdroje.
- **Vzory pro souběžnou správu více zdrojů současně** - popisují pravidla pro práci s větším počtem zdrojů v rámci jedné úlohy.

Vzory pro práci se zdroji sice lze obecně namodelovat s použitím prvků standardu BPMN, v jejich použití se ale skrývá mnoho vnitřních pravidel, které při použití dostupných prvků standardu lze vyjádřit jen s využitím semiformálních nebo neformálních metod. Podobné vlastnosti má také skupina vzorů pro práci s daty, která je obsahem následující kapitoly. Podrobný popis vzorů pro práci se zdroji [1][cit.16.3.2014] je dostupný z WWW,
$< http://www.workflowpatterns.com/patterns/resource/ >$.

3.4 Vzory pro práci s daty

Z pohledu práce s daty v průběhu zpracování byznys procesu se dají vyčlenit 4 skupiny, rozdělené podle charakteristik samotných dat nebo prostředí, ve kterém komunikace probíhá:

1. **Viditelnost dat** - jaké mají možnosti přístupu k datům prvky procesu.
2. **Interakce mezi daty** - způsob, jakým data komunikují s prvky procesu.
3. **Přenos dat** - zaměřuje se na způsob, jak jsou data předávána přes rozhraní mezi jednotlivými prvky procesu.
4. **Mapování cesty při komunikaci s daty** - do této skupiny se zařazují všechny možnosti, jak je ovlivněn samotný proces v okamžiku, kdy dochází k interakci mezi prvky a daty.

Reprezentace dat může obsahovat jednoduché datové typy jako je integer, float, boolean, date, time, nebo výčtový typ enumeration. Textové řetězce mají souhrnně datový typ string. Všechny základní datové typy lze řadit do složeného datového typu array. Jednotlivé úlohy si mohou pak předávat jak hodnoty proměnných, tak odkazy na proměnné. Vzory pro práci s daty se dále dělí na skupiny:

- **Vzory pro viditelnost dat** - podobně jako zdroje, i data mohou mít definovány úrovně přístupu pro určité úlohy a hodnoty.
- **Vzory pro výměna dat** - obsahuje vzory, popisující možnosti výměny datových elementů mezi úlohami a to jak pro data definována v rámci vnitřního uspořádání procesu, tak data získaná z vnějšího prostředí.
- **Vzory pro přenos dat** - řeší události, které se mohou vyskytnout během předávání dat mezi úlohami, případně cílenou úpravu dat během přenosu.
- **Vzory pro směrování dat** - zajímají se o předpoklady, které je nutné splnit před tím, než jsou datové elementy využity, případně způsob jejich využití na základě aktivované události.

Podrobný popis vzorů pro práci s daty [1][cit.16.3.2014] je dostupný z WWW, $< http://www.workflowpatterns.com/patterns/data/ >$.

3.5 Vzory pro zpracování výjimek

Způsob ošetření vyjímečných stavů v průběhu zpracování procesů zapracoval RUSSELL [13] do hodnotícího rámce PAIS (process-aware information system), jehož základ je založen na procesních vzorech. Pokud dojde k selhání provádění některého z prvků procesu, kromě zpracování řešení zotavení z chyby pro samotný prvek rozšiřuje pozornost také na způsob ovlivnění chybové stavu pro ostatní prvky, které jsou problémovou částí více či méně ovlivněny. Selhání prvku v procesu může dojít celou řadou způsobů, kde

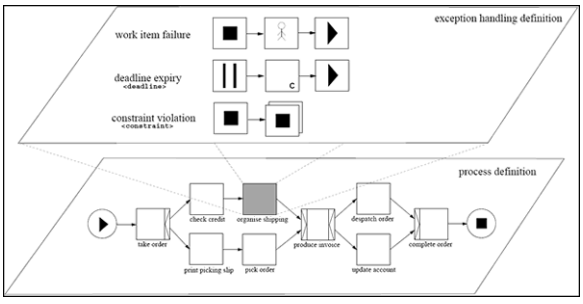
každá příčina následně vyžaduje jiný přístup pro zotavení se z výjimečného stavu. Může se jednat o chybu během provádění procesu v daném prvku, kde se kromě chyby v samotné logice procesu může jednat také o externí příčinu, jako je závada v software, hardware, cílené přerušení vykonávání procesu uživatelem, nebo překročení systémových omezení. Dalším problémem, který může nastat je chybná synchronizace s ostatními zdroji nebo objekty, kde může proces přistupovat k již neplatným datům, z důvodu chybné časové synchronizace je proces ukončen před svým provedením, nebo ostatní objekty, na které proces navazuje nejsou dokončeny v kompletním stavu. Podobný případ nastává, kdy proces potřebuje pro své provedení přístup k určitým zdrojům, které nejsou v požadovanou dobu k dispozici. Provádění procesu může být také přerušeno nenadálou událostí, které se proces nemusí přímo účastnit, ale která ovlivní ostatní objekty na proces navazující.
Pro každý prvek, který má být ošetřen pro zotavení se z výjimky se vytváří speciální sady kroků, kde každá sada je určena pro určitý výčet výjimek, které mohou nastat. Sady se postupně skládají ve třech fázích:

1. **Zpracování výjimky pro prvek procesu v návaznosti na zdroje, které prvek využívá** - výsledkem zpracování je rozhodnutí, zda může za určitých omezujících podmínek úloha dále pokračovat anebo je je chyba natolik kritická, že je úloha i s případnými již přidělenými zdroji zrušena nebo nahrazena.

2. **Dopad na instanci procesu** - selhání funkce jedné úlohy může mít za následek řetězovou reakci, která pak ovlivní funkci celého procesu. V tomto kroku se rozhodne, zda může celý proces dále pokračovat anebo to již není možné.

3. **Zotavení** - na základě informací z předešlých kroků se pokračuje dále beze změn, nebo dojde k návratu procesu do stavu, před kterým chyba nastala, nebo se problémové části procesu nahradí alternativním řešením.

Příklad, jak může být řešeno ošetření výjimky je na obrázku č. 3 [13]. V tomto příkladě se v případě selhání úlohy *organise shopping* podle druhu selhání provede jedna ze tří sad v rámci definice případu užití *exception handling definition*. Model procesu, graficky znázorněný podle standardu BPMN doplňuje PAIS svými vlastními grafickými prvky, zavedenými jen v rámci tohoto rámce.



Obrázek 3: Příklad provedení vybrané sady při výjimce během zpracování procesu

3.6 Prezentační vzory

Prezentační vzory mají za cíl pracovat především se syntaxí modelovacího nástroje, zatímco ostatní skupiny procesních vzorů se zaměřují na sémantiku procesu. Použití skupiny prezentačních vzorů přichází v úvahu u případů, kdy se model stává velmi rozsáhlým a je čím dál více obtížnější udržet konzistenci obsahu a formy. Jejich úkolem je pak model přehledně rozčlenit, zjednodušit, nebo jinak změnit jeho strukturu. Rodina prezentačních vzorů se dělí na dvě zcela odlišné skupiny:

- **Abstract Syntax** - zaměřuje se na formální vyjádření prvků procesu a vztahy mezi nimi. Vzory v této skupině lze ještě dále rozdělit na ty, které modifikují model a na ty, které mění použitý jazyk.
- **Concrete Syntax** - pracuje s vizuálním vyjádřením procesu, jako je zobrazení symbolů, použité barvy v modelu, umístění prvků na pozicích.

Podrobný popis prezentačních vzorů [1][cit.16.3.2014] je dostupný z WWW, $< http://www.workflowpatterns.com/patterns/presentation/ >$.

3.7 Použité prvky pro modelování

Při grafickém vyjádření vzorů se pro tento účel nevyužijí všechny prvky standardu BPMN, ale jen malá část z celkového počtu možných prvků. Na obrázku č. 4 je přehled prvků, které se pro modelování použijí přímo. Při použití specializovaného vyjádření prvku, jako je další členění typu úlohy, začátku procesu nebo konce procesu se provede substituce na abstraktní vyjádření typu.

Obrázek 4: Výběr grafických prvků BPMN použitý pro modelování vzorů

4 Vzory v modelovacích systémech

Procesní vzory byly definovány na základě pozorování jejich výskytu v široké množině modelovacích systémů. V současné době (k datu 26.4.2014) není takový systém, který by podporoval všechny procesní vzory popsané iniciativou WPI současně. Podpora pro procesní vzory se dá v zásadě rozdělit do několika skupin (seřazené podle kvality podpory od nejlepší varianty, po nejhorší):

- Vzor je podporován přímo, to znamená, že mohu vybrat z nabídky v menu systému jeho reprezentaci a dosadit ji do modelu,
- Systém vzor nezná a je nutné ho poskládat z dostupných modelovacích prvků,
- Systém vzor nezná a jeho vykreslení v modelu s pomocí dostupných prvků je velmi komplikované a nepřesné,
- Vzor nelze v modelu s pomocí dostupných funkcí vůbec realizovat.

V následující tabulce je přehled podpory ve známých modelovacích systémech pro skupiny vzorů, souhrnně pro všechny kategorie (komerční, open-source atd.) podle údajů dostupných ze stránek iniciativy WPI. Součástí tabulky je také určení, zda je možné pro danou skupinu použít algoritmus detekce vzorů pracující s modelem ve formátu BPMN, který je předmětem této práce. Do této skupiny nebudou spadat vzory, které obsahují potřebu použití vnitřní netriviální logiky, kterou nelze znázornit jen s použitím grafických prvků standardu BPMN.
V přehledu nejsou zahrnuty vzory pro správu výjimek, které mají svůj vlastní modelovací standard, vymykající se ze standardu BPMN. Také jsou z něj vyjmuty prezentační vzory, které jsou založeny především na myšlenkové logice použití vzoru a nelze je vyjádřit pomocí grafických prvků BPMN.

Hodnoty v následující tabulce jsou uvedené v procentech, s výjimkou sloupce určení použití pro detekci vzorů.

Skupina vzorů	**Přímá podpora**	**Lze modelovat**	**Nelze modelovat, nebo nepřesně**	**Použití pro detekci vzorů**
Základní řídící vzory	100	-	-	Ano
Pokročilé vzory pro rozdělení toku a synchronizaci	33	19	48	Ano
Vzory pro správu více instancí procesů	29	6	65	Částečně
Stavové vzory	28	10	62	Částečně
Iterační vzory	51	0	49	Ano
Vzory pro ukončení procesu	72	0	28	Ano
Vzory pro správu zahájení procesů	47	6	47	Ano
Vzory pro vytváření zdrojů	37	5	58	Velmi obtížně
Vzory pro alokování zdrojů	28	8	64	Velmi obtížně
Vzory pro využívání zdrojů	26	0	74	Velmi obtížně
Vzory pro správu využití zdrojů	20	3	77	Velmi obtížně
Vzory pro správu zahájení procesu při události	23	0	77	Velmi obtížně
Vzory pro správu viditelnosti zdrojů	0	12	88	Velmi obtížně
Vzory pro správu více zdrojů současně	45	0	55	Ne
Vzory pro správu viditelnosti dat	39	17	44	Velmi obtížně
Vzory pro datovou komunikaci	26	8	66	Velmi obtížně
Vzory pro správu směrování dat	45	8	47	Velmi obtížně

Tabulka 5: Podpora v modelovacích systémech pro rodiny vzorů

5 Detekce vzorů v modelu

Cílem detekce vzorů v modelu je identifikovat v libovolném modelu jakékoliv velikosti konkrétní procesní vzory, tím je myšleno jejich abstraktní vyjádření pro řešení konkrétního problému v modelu. Základním předpokladem pro identifikaci vzorů v modelu je shodná formální definice jak pro model, tak pro vzory. Model procesu vyjádřený v BPMN si lze představit jako orientovaný graf, kde spojovací objekty tvoří hrany grafu a ostatní objekty vrcholy. Orientovaným grafem rozumíme uspořádanou dvojici $G = (V, E)$, kde V je množina vrcholů a množina orientovaných hran je $E \subseteq V \times V$ [5]. Většinu vzorů je možné v generickém tvaru zapsat do některé ze semiformální nebo formální formy (BPMN, UML, Petriho sítě) a následně převést jejich reprezentaci na orientovaný graf. Jestliže je model a návrhový vzor reprezentován jako graf, pak je problém převeden na úlohu hledání podgrafu v grafu. V případě porovnání menšího vzoru s větším modelem můžeme zkoumat, zda po odstranění některých hran a vrcholů z většího grafu existuje izomorfismus mezi oběma grafy.

Proces ověření shody struktury grafů je běžně nazýván pojmem *"graph matching"*. Metody, které se touto problematikou zabývají se dělí na ty, které vyhledávají přesnou shodu porovnávaných grafů (řadí se mezi NP-úplné problémy) a na ty, které dovolují určitou míru odchylku shody [6]. Jestliže je možné pro každý vrchol podgrafu najít obraz ve zkoumaném grafu s totožným propojením vrcholů hranami, pak je tento způsob izomorfismu nazýván jako *"homomorfismus"* [10]. Většina metod využívá pro definici grafu a podgrafu formálního vyjádření dle definice:

Definice 5.1 *Formálně je ohodnocený graf vyjádřen uspořádanou čtveřicí* (V, E, ω, ν)*: [6]*

- ***V** je neprázdná konečná množina vrcholů,*
- ***E** je množina hran, kde* $E \subseteq V \times V$*,*
- ω *je incidenční zobrazení* $f : E \to \mathbb{R}$*, tedy funkce, která každé hraně přiřadí unikátní pojmenování.*
- ν *je incidenční zobrazení* $f : V \to \mathbb{R}$*, tedy funkce, která každému vrcholu přiřadí unikátní pojmenování.*

Definice 5.2 *Nechť je dán graf* $\mathrm{g}_1 = (\mathrm{V}_1, \mathrm{E}_1, \omega_1, \nu_1)$ *a graf* $\mathrm{g}_2 = (\mathrm{V}_2, \mathrm{E}_2, \omega_2, \nu_2)$*. Graf* g_1 *je podgrafem* g_2*, píšeme* $\mathrm{g}_1 \subseteq \mathrm{g}_2$*, jestliže: [6]*

- $\mathrm{V}_1 \subseteq \mathrm{V}_2$*,*
- $\mathrm{E}_1 = \mathrm{E}_2 \bigcap (\mathrm{V}_1 \times \mathrm{V}_1)$*,*
- $\omega_1 (u) = \omega_2 (u)$ *pro všechna* $u \in \mathrm{V}_1$*.*
- $\nu_1 (u, v) = \nu_2 (u, v)$ *pro všechna* $u, v \in \mathrm{E}_1$*..*

5.1 Metody ověření izomorfismu grafu a podgrafu

5.1.1 Technika založená na prohledávání stromu

Podle COOK[6] je na začátku hledání založen graf s výsledkem, který je při zahájení algoritmu prázdný. Následně jsou iterativně prohledávány zkoumané grafy a kdykoliv je nalezena shoda dvou propojených vrcholů v obou grafech, je tento pár vrcholů přenesen do grafu s výsledkem. Ve chvíli, kdy jsou oba grafy kompletně zkontrolovány, v grafu výsledku se v případě přítomnosti izomorfního podgrafu v grafu nachází graf, který je shodný s hledaným vzorem. V opačném případě je v grafu výsledku přítomna jen část hledaného grafu nebo zůstane prázdný při nenalezení ani jedné shody. Zřejmá nevýhoda tohoto postupu je výpočetní komplexnost algoritmu, která exponenciálně roste v závislosti na počtu hran a vrcholů obou grafů.

5.1.2 Výčtový algoritmus pro určení isomorfismu podgrafu

V roce 1976 uvedl ULLMANN [8] algoritmus pro porovnávání isomorfismu grafů založený na principu porovnávání stupňů vrcholů v maticích sousednosti. Princip je následující - pro dva grafy G_A a G_B je definována jejich matice sousednosti A a B. Je dána matice M, která má počet řádků roven počtu vrcholů matice A a počet sloupců odpovídá počtu vrcholů matice B. Postupně se projdou všechny prvky matice $M_{x,y}$ a ověří se, jaký je stupeň uzlu prvku A_y v porovnání se stupněm uzlu prvku B_x. Jestliže je stupeň uzlu prvku A_y vyšší nebo stejný jako u prvku A_y, prvek matice $M_{x,y}$ bude mít hodnotu 1, jinak má hodnotu 0.

Po naplnění matice M se ověří pro každý prvek $M_{x,y}$, který má hodnotu 1, zda každý sousední vrchol grafu G_A vyjádřený v matici A na pozici x má odpovídají sousední vrchol pro graf G_B vyjádřený maticí B na pozici y. Procházení matice M se provádí po sloupcích, kdy po každém ověření sloupce (počínaje prvním řádkem) se přejde na další řádek. Každý úspěšně ověřený prvek ve sloupci matice M zachová v samostatné instanci, která se testuje v tomto uloženém stavu na dalším řádku. Jestliže v některém řádku test selže, instance matice se odstraní. Po ověření posledního řádku, ty instance matic, které nebyly odstraněny, obsahují matici sousednosti izomorfního grafu ke grafu vzoru.

5.1.3 VF2 algoritmus

Jedná se o rekurzívní algoritmus, který prochází postupně vrcholy zkoumaného grafu a souběžně vrcholy vzoru, jehož autorem je CORDELLA [9]. V každém kroku, jestliže propojení dvou vrcholů je shodné u obou grafů, je tato dvojice vložena do mapovací kolekce M. Pokud funkce narazí na vrchol, který není obsažen ve vzoru pak se provede krok zpět. Jestliže je zkoumán vrchol, ze kterého vede více hran, funkce se rekurzivně vydá po všech hranách, které z vrcholu vedou. Jakmile jsou všechy procesy hledání ukončeny (všechny procesy dorazily do vrcholu, ze kterého nevede žádná další hrana), v kolekci M se nachází výsledek porovnání, který je nebo není shodný s grafem vzoru.

5.1.4 Nauty

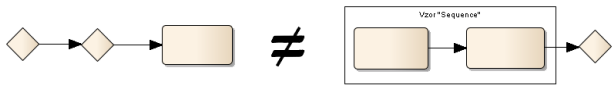
Algoritmus založený na teorii skupin, uvedený v roce 1981 Brendanem D. McKayem [11], je považován za jeden z nejrychlejších algoritmu pro hledání izomorfismu mezi grafy [10]. Základním principem algoritmu je rozdělení vrcholů do skupin podle jejich stupně. Z těchto skupin se pak vytváří vyhledávací strom, od vrcholů s nejvyšším stupněm sousednosti až po vrcholy, které mají stupeň roven hodnotě 1. Vygenerované stromy obou grafů pak stačí jen porovnat do jaké míry jsou ekvivalentní.

5.1.5 Ostatní metody

Kromě metod, které pracují přímo s prvky grafu lze graf dekomponovat na matici sousednosti, Laplaceovou matici, maticí vzdálenosti nebo matici incidence. V takovém případě se pak úloha hledání isomorfismu mezi grafy převede na operace s maticemi [12].

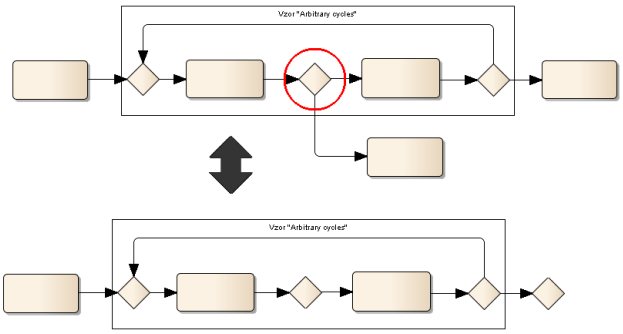
5.1.6 Výběr vhodné metody

I když uvedené algoritmy vypadají lákavě, hledání vzoru v byznys modelu má řadu omezení, které výběr metody značně ztíží. Na obrázku č. 5 jsou znázorněný dva podobné modely, kde obsažený vzor nemá svůj obraz v druhém modelu. Pokud ale na tyto modely nahlížíme jako na dva grafy, jsou isomorfní.



Obrázek 5: Porovnání dvou modelů s obsaženým vzorem Sequence

Omezení v použití mají také algoritmy, které pracují s hodnotou stupně vrcholů. Na obrázku č. 6 jsou dva modely, kde i když každý z nich má jiný počet výstupních hran, tak jsou k sobě ekvivalentní. Tento problém je nejvíce patrný u vzorů rozdělujících a spojujících tok a také u některých prezentačních vzorů.



Obrázek 6: Porovnání dvou modelů se vzorem Arbitrary cycles

S přihlédnutím k těmto omezením jsem si pro řešení úlohy srovnání byznys modelu a vzoru vybral metodu, založenou na hledání isomorfismu mezi dvěma incidenčními maticemi, kterou omezení vypsané v této kapitole nezatěžuje. Dodatečně je jen nutné u této metody evidovat použitý typ prvku pro každý řádek modelu.

5.2 Reprezentace grafu pomocí matice incidence

Z definice grafu vyplývá, že jeho základní reprezentace jsou množiny hran, vrcholů a jejich identifikační hodnoty. Hrany a vrcholy grafu lze převést na matici incidence přímo, identifikační hodnoty hran i vrcholů se musí evidovat formou vhodnou pro zpracování kolekcí, jako je pole objektů nebo kolekce objektů. Formální vyjádření matice incidence je podle definice:

Definice 5.3 ***Incidenční matice*** *grafu G je matice S s* $|V| = n$ *řádky a* $|\omega| = m$ *sloupci. Každému vrcholu grafu G odpovídá jeden řádek matice a každé hraně jeden sloupec matice. Její prvky* S_{mn} *jsou dány předpisem: [3]*

$$S_{mn} = \begin{cases} -1 & \textit{jestliže hrana } m \textit{ končí ve vrcholu } n \\ 1 & \textit{jestliže hrana } m \textit{ vychází z vrcholu } n \\ 0 & \textit{hrana } m \textit{ s vrcholem } n \textit{ není sousední} \end{cases} \tag{1}$$

5.3 Postup hledání podmatice vzoru v matici modelu

Nejprve jsou všechny procesní vzory a zkoumaný model převedeny ma incidenční matice. Matice vzoru si označme V_j, matici modelu M. Incidenční zobrazení ω pro každý graf obsahuje množinu typů elementů každého objektu v grafu pomocí funkce $f : P_m = x, x \in$ ⟨množina všech typů elementů, převzaná z notace BPMN⟩ a odpovídá řádkům matice. Pro matici vzoru to bude zobrazení ω_v, pro matici modelu ω_m. Incidenční zobrazení ν je množina unikátních identifikátorů vrcholů, tvořených řetězcem čísel, znaků abecedy a speciálních znaků a odpovídá sloupcům matice. Pro matici vzoru se jedná o zobrazení ν_v, pro matici modelu ν_m. Vzory se v modelu hledají postupně podle své velikosti, od nejmenších (atomických) až po nejkomplexnější. Kdykoliv je v matici modelu nalezena podmatice vzoru, všechny objekty v modelu, které vzoru odpovídají se z modelu vyjmou a místo nich se dosadí objekt základní úlohy. Hledání se pak znovu opakuje od začátku. Ve chvíli, kdy již není v modelu nalezen žádný vzor, hledání se ukončí. Proměnná pořadí matice vzoru j se nastaví na první matici v množině matic vzorů. Na začátku každého hledání pro každý vzor se v matici modelu M nachází reprezentace celého modelu s výjimkou již nalezených vzorů, které jsou nahrazeny objektem základní úlohy.

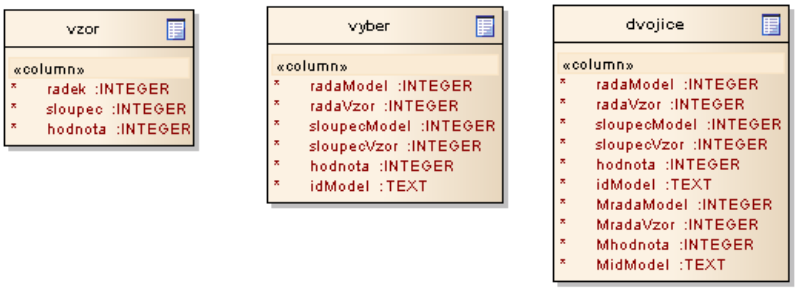
Asymptotická složitost algoritmu má exponenciální růst, proto jsou data v části porovnávání matic přenesena na embedded databázový server, kde je úloha převedena na výběr z relace kartézského součinu sloupců a řádků matice vzoru podle vazby na odpovídající sloupce a řádky vzoru. S provedenou úpravou má složitost téměř lineární růst.

Popis algoritmu:

1. Algoritmus projde všechny řádky matice M a ověří, zda pro každý typ prvku zkoumaného řádku $\omega(y) \in \omega_m$ existuje minimálně jeden totožný typ prvku v incidenčním zobrazení ω_{v_j}. Jestliže takový typ neexistuje, řádek je z matice M odstraněn. Po dokončení tohoto kroku bude ω_m obsahovat jen ty typy prvků, které obsahuje také ω_{v_j}.

2. Iterativně jsou porovnány všechny prvky ve sloupcích matice M a V_j. Jestliže pro hodnotu prvku $M_{(x,y)}$ neexistuje ve sloupci $V_{j_{(x)}}$ odpovídající hodnota anebo je jeho hodnota 0, je prvek z matice M odstraněn. Po dokončení porovnání prvků, jestliže se na začátku porovnávání v matici M nacházel obraz vzoru V_j pro hodnoty prvků 1 nebo -1, pak platí $M \simeq V_j$.

3. Hodnoty matice vzoru V_j a matice M jsou přeneseny do databázových tabulek $vzor$ a $vyber$, definovaných podle EER diagramu na obrázku č. 3. Atribut $idModel$ obsahuje jednoznačnou identifikaci prvku v modelu a je použit po identifikaci vzoru pro jeho vyjmutí ze základního modelu.



Obrázek 7: EER diagram tabulek vzoru, výběru z modelu a dvojic

4. Jelikož mám jistotu, že každá hrana má vždy právě jeden začátek a jeden konec, pro každý sloupec výběru z modelu si vytvořím záznam v tabulce $dvojice$, který reprezentuje všechny informace o zkoumané hraně v návaznosti na identickou hranu v modelu.
   Atributy s prefixem M obsahují informace o prvku s hodnotou -1, ostatní atributy jsou určeny pro prvek s hodnotou 1. Oba prvky využívají jednu hodnotu sloupce, která je určená atributem $sloupecVzor$ a $sloupecModel$.

5. Posledním krokem je výběr z kartézského součinu záznamů z tabulky $dvojice$, kde omezující podmínky jsou shodné řádky a sloupce vzoru i výběru a zároveň pravidlo, že každý sloupec může být ve vzoru i modelu obsažen jen jednou. SQL příkaz je vzhledem k možnostem embedded SQLite databázi velmi zjednodušený a omezený, ve srovnání s plnohodnotnými relačními databázemi:

```
select dv.idModel as idModel1, dx.idModel as idModel2, dv.MidModel as idModel3, dx.
    MidModel as idModel4
from dvojice dv, dvojice dx
where dv.sloupecVzor!=dx.sloupecVzor and dv.sloupecModel!=dx.sloupecModel
and dv.radaVzor=dx.radaVzor and dv.radaModel=dx.radaModel
group by dv.radaVzor, dv.MradaVzor, dv.sloupecVzor
union
select dv.idModel, dx.idModel, dv.MidModel, dx.MidModel from dvojice dv, dvojice dx
where dv.sloupecVzor!=dx.sloupecVzor and dv.sloupecModel!=dx.sloupecModel
and dv.MradaVzor=dx.MradaVzor and dv.MradaModel=dx.MradaModel
group by dv.radaVzor, dv.MradaVzor, dv.sloupecVzor
union
select dv.idModel, dx.idModel, dv.MidModel, dx.MidModel from dvojice dv, dvojice dx
where dv.sloupecVzor!=dx.sloupecVzor and dv.sloupecModel!=dx.sloupecModel
and dv.radaVzor=dx.MradaVzor and dv.radaModel=dx.MradaModel
group by dv.radaVzor, dv.MradaVzor, dv.sloupecVzor
union
select dv.idModel, dx.idModel, dv.MidModel, dx.MidModel from dvojice dv, dvojice dx
where dv.sloupecVzor!=dx.sloupecVzor and dv.sloupecModel!=dx.sloupecModel
and dv.MradaVzor=dx.radaVzor and dv.MradaModel=dx.radaModel
group by dv.radaVzor, dv.MradaVzor, dv.sloupecVzor
```

Výpis 1: SQL příkaz pro výběr prvků modelu podle odpovídajících sloupců a řádků vzoru

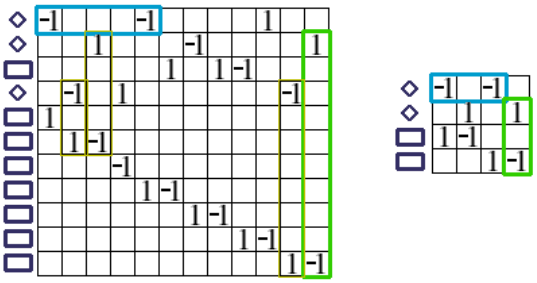
Získané atributy $idModel1$ až $idModel4$ obsahují výčet unikátních identifikátorů prvků výběru, které budou z modelu odstraněny (obsahují obraz vzoru).

Výjimku tvoří vzor $Sequence$, který je tvořen jen jednou hranou a nelze u něj provést porovnání více dvojic ve sloupcích. Pro tento případ je určen jiný SQL příkaz:

```
select distinct(dv.idModel) as idModel, dv.MidModel as MidModel from dvojice dv, vzor vz
where ((dv.hodnota=vz.hodnota and dv.radaVzor=vz.radek)
or (dv.Mhodnota=vz.hodnota and dv.MradaVzor=vz.radek)) and dv.sloupecVzor=vz.sloupec
group by dv.sloupecVzor, dv.radaVzor, dv.MradaVzor, vz.radek, vz.sloupec, vz.hodnota
```

Výpis 2: SQL příkaz pro vzory obsahující jen jednu hranu

Na následujícím obrázku je příklad porovnání mezi vzorem $Multi\ Merge$ a modelem, který tento vzor obsahuje.



Obrázek 8: Ukázka matice vzoru Multi Merge a modelu, který tento vzor obsahuje

6. Z matice M se vybere takový řádek, který neobsahuje hodnotu -1 u kterého platí $idModel \in (idModel1, idModel2, idModel3, idModel4)$. Do prvků tohoto řádku se z ostatních řádků matice M u kterých platí $idModel \in (idModel1, idModel2, idModel3, idModel4)$ přepíší všechny hodnoty 1, které se nachází ve stejných sloupcích a pak se tyto řádky z matice M odstraní. Typ elementu přiřazenému k řádku, který v matici M jako jediný zůstal, se nastaví na typ elementu základní (abstraktní) úlohy.

7. Proměnná j se navýší o jedna. Pokud existuje matice vzoru V_{j+1} přejde se na krok 1. a hledání se znovu opakuje, jinak se zpracování algoritmu ukončí.

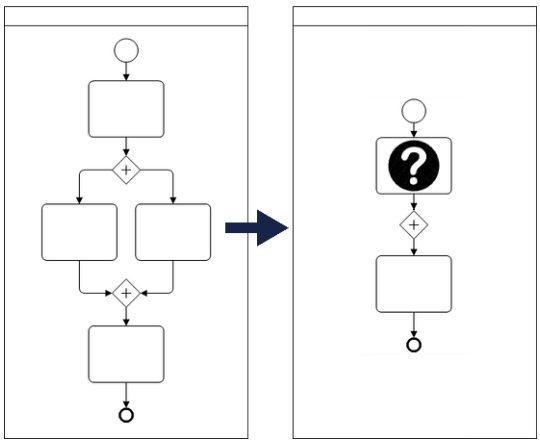
5.4 Omezující vlastnosti algoritmu

Bezproblémová funkce je zaručena pro vzory, obsahující maximálně 5 hran, které nejsou parciálně zaměnitelné s jinou částí modelu. U větších vzorů se může vyskytnout problém s duplicitním označením některého sloupce (platí pro vzory, které mají počet shodných typů prvků vyšší než 2). Dalším známým problémem je skutečnost, že algoritmus nemusí rozpoznat správné umístění v modelu dalších podmatic z podmatice - typickým příkladem je vzor $Structured\ Synchronizing\ Merge$, který je složen ze vzorů $Exclusive\ Choice$ a $Synchronizing\ Merge$. Vzhledem k těmto skutečnostem se algoritmus řadí do kategorie s určitou odchylkou shody u porovnávaných grafů.

Ze zkušeností z praktického používání vzorů při modelování se vyplatí hledat především vzory s maximálním počtem hran 4. Při použití vzoru s větším počtem hran analytik vzor většinou zdeformuje tak, aby s jeho pomocí dokázal co nejlépe namodelovat požadovaný byznys proces. Proces nebere žádný ohled na syntaxi vzoru a musí být zachycen přesně tak, jak probíhá v reálném životě.

5.5 Výjimky pro nahrazení vzorů základním objektem

Při odstraňování prvků v modelu podle nalezeného vzoru nelze mechanicky každou sestavu prvků volně nahradit objektem základní úlohy. Pokud mají být některé prvky z modelu vyjmuty, nesmí se stát, že se ztratí interpretace modelovaného procesu a kontext ztratí smysl. Příklad, jak k takové situaci může dojít je znázorněn na obrázku č. 9, kde se nachází vzájemně propojené vzory *Parallel Split* a *Synchronization*. Při odstranění prvků přináležících ke vzoru *Parallel Split* přestane být zřejmé, na základě jaké podmínky má být spojení procesů synchronizováno. Z toho důvodu jsou při hledání vzorů v modelu použity jen takové vzory, které ve své podstatě při nahrazení v modelu nezpůsobí konflikt v interpretaci procesu.



Obrázek 9: Ztráta informace o interpretaci brány

6 Testovací aplikace

Pro detekci procesních vzorů ve zkoumaných byznys procesech jsem vytvořil desktopovou aplikaci, s možností evidence kolekce vybraných vzorů. Aplikace byla vytvořena na základě formální analýzy, která je obsahem dalších podkapitol. Vzhledem k malému rozsahu požadavků jsem pro vývojový proces použil sekvenční vodopádový model.

6.1 Specifikace požadavků a scénáře případů užití

Aplikace by měla po vložení byznys modelu libovolné velikosti provést srovnání s dostupnými vzory v aplikaci, zaznamenat jejich výskyt v modelu a po nalezení všech výskytů zobrazit souhrnnou sestavu o provedené operaci. Uživatel může vkládat do aplikace vzory a modely s pomocí funkce exportu dat programu Enterprise Architect verze 10, který poskytuje komfortní a ucelené prostředí pro vytváření a úpravu modelů podle standardu BPMN. Přenos dat mezi aplikacemi se bude provádět textovými soubory s použitím značkovacího jazyka XML. V aplikaci bude k dispozici grafický náhled pro zpracovávané vzory i modely.

Název: Hlavní nabídka
ID: DV00001
Vstupní podmínky: Nejsou
Hlavní scénář:

1. Systém: Jestliže neexistuje žádný model a/nebo vzor v paměti, systém načte obsah soubor XML a inicializuje seznam modelů a vzorů.
2. Systém: Čeká na zadání příkazu.

Alternativní scénář:

1. Uživatel: Zadá příkaz ”Byznys model”.
2. Systém: Přejde na případ užití ID: DV01001 s vybraným typem modelu ”byznys model”.

Alternativní scénář:

1. Uživatel: Zadá příkaz ”Procesní vzory”.
2. Systém: Přejde na případ užití ID: DV01001 s vybraným typem modelu ”procesní vzor”.

Alternativní scénář:

1. Uživatel: Zadá příkaz ”Import EA”.
2. Systém: Přejde na případ užití ID: DV01002.

Název: Byznys model / Procesní vzor
ID: DV02001
Vstupní podmínky: Musí být vybrán typ modelu
Hlavní scénář:

1. Systém: Jestliže uživatel měl zobrazen některý model před odchodem z tohoto případu užití, tento model zobrazí na pracovní ploše, jinak má uživatel k dispozici prázdnou plochu. Zobrazí seznam modelů, se kterými může uživatel pracovat.
2. Uživatel: Může vybrat model ze seznamu, zadat příkaz "Vyhledat vzory"nebo "Smazat model".

Alternativní scénář:
Vstupní podmínky: Model byl načten ve scénáři ID DV02001 a není uložen v systému.

1. Uživatel: Zadal příkaz "Uložit model".
2. Systém: Nabídne možnost zadat nový název modelu, nebo vybrat stávající model ze seznamu.
3. Uživatel: Vloží nový název modelu, nebo vybere některý model ze seznamu. Název musí být zadán nebo vybrán model, jinak scénář dále nepokračuje.
4. Systém: Uloží nový, nebo přepíše vybraný model do souboru XML podle zvoleného typu jako byznys model, nebo procesní vzor.

Alternativní scénář:
Vstupní podmínky: Uživatel vybral model ze seznamu byznys modelů, nebo procesních vzorů.

1. Uživatel: Zadal příkaz "Smazat model".
2. Systém: Zobrazí dotaz pro potvrzení smazání modelu.
3. Uživatel: Potvrdí nebo zamítne dotaz.
4. Systém: Pokud je dotaz potvrzen, vymaže model z paměti.

Alternativní scénář:
Vstupní podmínky: Vybraný typ modelu musí být byznys model, jinak tento scénář nebude uživateli k dispozici.

1. Uživatel: Zadá příkaz "Vyhledat vzory".
2. Systém: Jestliže v modelu na pracovní ploše se nenacházejí alespoň dva propojené elementy, proces dále nepokračuje a uživatel je o tom informován.
3. Systém: Provede algoritmus pro vyhledání vzorů v aktivním modelu.
4. Systém: Zobrazí souhrnnou zprávu z provedené detekce vzorů v modelu.

Název: Import EA
ID: DV02001
Vstupní podmínky: Nejsou
Hlavní scénář:

1. Systém: Zobrazí formulář pro výběr souboru modelu nebo vzoru.
2. Uživatel: Vybere soubor modelu nebo vzoru.

Alternativní scénář:

1. Uživatel: Vybral soubor modelu.
2. Systém: Načte soubor do paměti a zpracuje jej. Zobrazí hlášení o úspěchu nebo selhání zpracování souboru. Úspěšně načtený model se stane aktivním modelem. Přejde na případ užití DV02001 s vybraným typem modelu "byznys model".

Alternativní scénář:

1. Uživatel: Vybral soubor vzoru.
2. Systém: Načte soubor do paměti a zpracuje jej. Zobrazí hlášení o úspěchu nebo selhání zpracování souboru. Úspěšně načtený model se stane aktivním modelem. Přejde na případ užití DV02001 s vybraným typem modelu "procesní vzor".

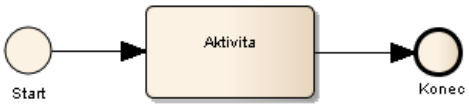
6.2 Datové rozhraní aplikace Enterprise Architect

Enterprise Architect poskytuje širokou škálů formátů, ve kterých je možné vytvořený model exportovat pro další použití v jiných aplikacích. Jako nejvhodnější formát pro účely použití v testovací aplikaci je XML s typem exportu "BPMN 2.0 XML". Základní informace o celém modelu jsou v root elementu $\langle bpmn : definitions \rangle$. Informace o jednotlivých prvcích jsou rozděleny na dvě části, $\langle bpmn : process \rangle$ obsahuje informace o obsažených prvcích v modelu a jejich provázání, $\langle bpmndi : BPMNDiagram \rangle$ je souhrnem vlastností prvků, které se vztahují k zobrazení v aplikaci Enterprise Architect (pozice, typ elementu). Každý prvek v procesu má přidělený unikátní identifikátor (id), který je klíčový při rozlišení provázání prvků pomocí elementů $\langle bpmn : incoming \rangle$ a $\langle bpmn : outgoing \rangle$. Způsob zanoření elementů je znázorněn v příkladu 3, s výpisem souboru XML a jednoduchým modelem obsahující tři prvky. Pro rozlišení prvků v aplikaci na základě přidělených značek XML je součástí aplikace konfigurační soubor WorkflowPatterns.exe.config, který je možné upravovat na uživatelské úrovni.

Datové rozhraní není ve všech směrech ideální, při jeho používání v praxi jsem narazil na řadu nedostatků. Mezi nejpodstatnější patří skutečnost, že u starších verzí tohoto programu aplikace "zapomene"u některých prvků vyexportovat jejich pozici a rozměry. Také neposkytuje přesné mapování grafického znázornění cesty v asociaci mezi prvky, takže k dispozici je jen přímé propojení z jednoho prvku do druhého. Tyto nedostatky jen ztěžují grafickou vizualizaci modelu, u klíčových informací, které přímo specifikují použité prvky a jejich asociace jsem na žádný problém nenarazil.

```
<?xml version="1.0" encoding="windows-1252"?>
<bpmn:definitions id="EAPK_DE05D573_67CE30652F67" targetNamespace="www.sparxsystems.
    com.au/bpmn20" exporter="Sparx Systems">
  <bpmn:process id="EAID_DP000000_67CE30652F67">
    <bpmn:startEvent id="EAID_0D8B0B937AAE" name="Start">
      <bpmn:outgoing>EAID_16F4B9472668</bpmn:outgoing>
    </bpmn:startEvent>
    <bpmn:endEvent id="EAID_DD44F47F20DF" name="Konec">
      <bpmn:incoming>EAID_223A4BF6CC6F</bpmn:incoming>
    </bpmn:endEvent>
    <bpmn:task id="EAID_7942E2CF97C5" name="Aktivita">
      <bpmn:incoming>EAID_16F4B9472668</bpmn:incoming>
      <bpmn:outgoing>EAID_223A4BF6CC6F</bpmn:outgoing>
    </bpmn:task>
    <bpmn:sequenceFlow id="EAID_223A4BF6CC6F" sourceRef="EAID_7942E2CF97C5"
        targetRef="EAID_DD44F47F20DF"/>
    <bpmn:sequenceFlow id="EAID_16F4B9472668" sourceRef="EAID_0D8B0B937AAE"
        targetRef="EAID_7942E2CF97C5"/>
  </bpmn:process>
  <bpmndi:BPMNDiagram id="EAID_3662CB51_E638A76A00F0" name="BPMN 2.0 Model">
    <bpmndi:BPMNPlane id="EAID_PL000000_E638A76A00F0" bpmnElement="
        EAID_DP000000_67CE30652F67">
      <bpmndi:BPMNShape id="EAID_DO000000" bpmnElement="EAID_0D8B0B937AAE">
        <dc:Bounds x="140" y="137" width="30" height="30"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="EAID_DO000001" bpmnElement="EAID_7942E2CF97C5">
        <dc:Bounds x="231" y="123" width="110" height="60"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="EAID_DO000002" bpmnElement="EAID_DD44F47F20DF">
        <dc:Bounds x="415" y="138" width="30" height="30"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="EAID_DO000000" bpmnElement="EAID_223A4BF6CC6F">
        <di:waypoint x="341" y="153"/>
        <di:waypoint x="415" y="153"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="EAID_DO000001" bpmnElement="EAID_16F4B9472668">
        <di:waypoint x="170" y="152"/>
        <di:waypoint x="231" y="152"/>
      </bpmndi:BPMNEdge>
    </bpmndi:BPMNPlane>
  </bpmndi:BPMNDiagram>
</bpmn:definitions>
```

Výpis 3: Příklad exportu modelu ve formátu XML z aplikace Enterprise Architect



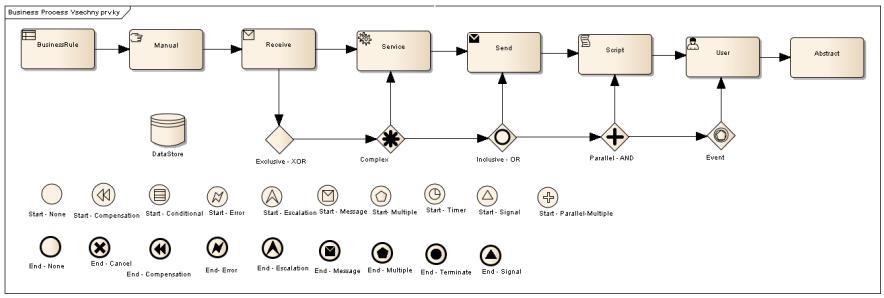
Obrázek 10: Exportovaný model, jak je zobrazen v aplikace Enterprise Architect

6.3 Kvalitativní požadavky

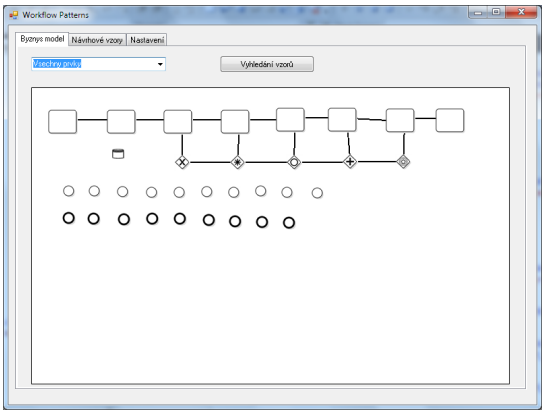
Systém bude implementován jako samostatná desktopová aplikace bez nutnosti instalace do systému s použitím .NET technologie. Operační systémy, na kterých bude možné aplikaci provozovat jsou Windows XP, Vista, 7, 8 s nainstalovaným .NET frameworkem verze 4.0. Data, vložená do aplikace budou persistována v souboru XML, který bude uložen ve stejném adresářovém umístění, jako samotná aplikace.

6.4 Ukázky výstupů zpracování testovaných dat

Pro korektní porovnávání modelů a vzorů, importované z aplikace Enterprise Architect, bylo prvním úkolem ověření datového rozhraní mezi oběma aplikacemi. K tomuto účelu jsem vytvořil abstraktní model, obsahující všechny potřebné prvky standardu BPMN, které se vyskytují v modelovaných procesech. Po odladění rozhraní bylo dosaženo shody mezi aplikacemi, jak je znázorněno na následujících ukázkách.



Obrázek 11: Model v aplikaci EA, obsahující výčet potřebných prvků standardu BPMN



Obrázek 12: Zobrazený model v testovací aplikaci, po importu z aplikace EA

Po ověření funkce následovala sada testů s různými skupinami vzorů a modelů, které jsou obsahem přílohy.

7 Závěr

V rámci mé diplomové práce jsem se měl seznámit s používanými postupy při detekci procesních vzorů v byznys modelech a jednu vybranou metodu realizovat ve vlastním aplikačním prostředí.

Výsledkem je aplikace, která dokáže pracovat s modely vytvořenými v prostředí aplikace Enterprise Architect a s pomocí algoritmu založeném na porovnávání dvou matic nalézt ve zkoumaném modelu vzory, pokud jsou v něm obsaženy. Ačkoliv se algoritmus řadí do kategorie přibližné metody detekce, testy prokázaly, že důkaz přítomnosti vzoru v modelu je dostatečně relevantní a vzor je cíleně rozpoznán. Užitečným krokem pro provedení algoritmu bylo převedení části výpočetní náročnosti na databázový server, který mnohonásobně urychlil proces porovnání matic, takže je možné v uspokojivém čase zpracovat pomocí aplikace i modely s větším počtem prvků.

Velkým osobním přínosem je pro mne seznámení se s širokou rodinou procesních vzorů, pro které jako analytik dokáži najít široké uplatnění. Aplikace, která je výsledkem této práce není pro mne jen testovací nástroj pro jedno použití, ale můžu ji využít ve své každodenní praxi, kdykoliv se budu sám sebe ptát, zda dělám věci správně.

Z pohledu dalšího vývoje projektu by bylo velmi užitečné vytvořit rozhraní pro import dat také z dalších modelovacích systémů, ve kterých se procesní vzory používají. V této práci jsou pro detekci vzorů v modelu použity jen řídící vzory, dalším krokem by mělo být rozšířit možnost detekce také pro ostatní skupiny procesních vzorů.

8 Reference

[1] W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros, *Workflow Patterns*. Dostupný z WWW: http://www.workflowpatterns.com/.

[2] Object Management Group, Inc., *Business Process Model and Notation, version 2.0*. Dostupný z WWW: http://www.omg.org/spec/BPMN/2.0, [cit. 24.10.2013].

[3] Mgr. Petr Kovář Ph.D., *Úvod do Teorie Grafů*. VŠB-TU Ostrava, 2012.

[4] Prof. Ing. Ivo Vondrák, CSc., *Úvod do softwarového inženýrství, verze 1.1*. VŠB-TU Ostrava, 2002.

[5] Doc. RNDr. Ing. Miloš Šeda, PhD., *Teorie grafů*. VUT FSI v Brně 2003.

[6] Diane J. Cook, Lavrence B. Holder, *Mining graph data*. Hoboken, ISBN 978-047-1731-900, 2007.

[7] Dietmar Kühl, *Design Patterns for the implementation of Graph Algorithms*. Technische Universität Berlin, 1996.

[8] J.R. Ullman, *An algorithm for subgraph isomorphism*. J. Assoc. Comput. Mach., 1976.

[9] L. P. Cordella, P. Foggia, C. Sansome, M. Vento, *A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs*. IEEE Trans. Patterns Anal. Mach. Intell., vol. 26, no. 10, 2004.

[10] D. Conte, P. Foggia, C. Sansome, M. Vento, *Thirty years of graph matching in pattern recognition*. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 18, no. 3, 2004.

[11] Brendan D. McKay, *Nauty User's guide*. Department of Computer Science Australian National University Canberra ACT 0200, Australia, 2009.

[12] Ali Kaveh, *Introduction to Graph Theory and Algebraic Graph Theory*. Springer Vienna, 2013.

[13] Nick Russell, W.M.P van der Aalst, A.H.M. ter Hofstede, *Exception Handling Patterns in Process-Aware Information Systems*. School of Information Systems, Queensland University of Technology.

[14] Nick Russell, A.H.M. ter Hofstede, David Edmong *Workflow Data Patterns*. Centre for Information Technology Innovation, Queensland University of Technology.

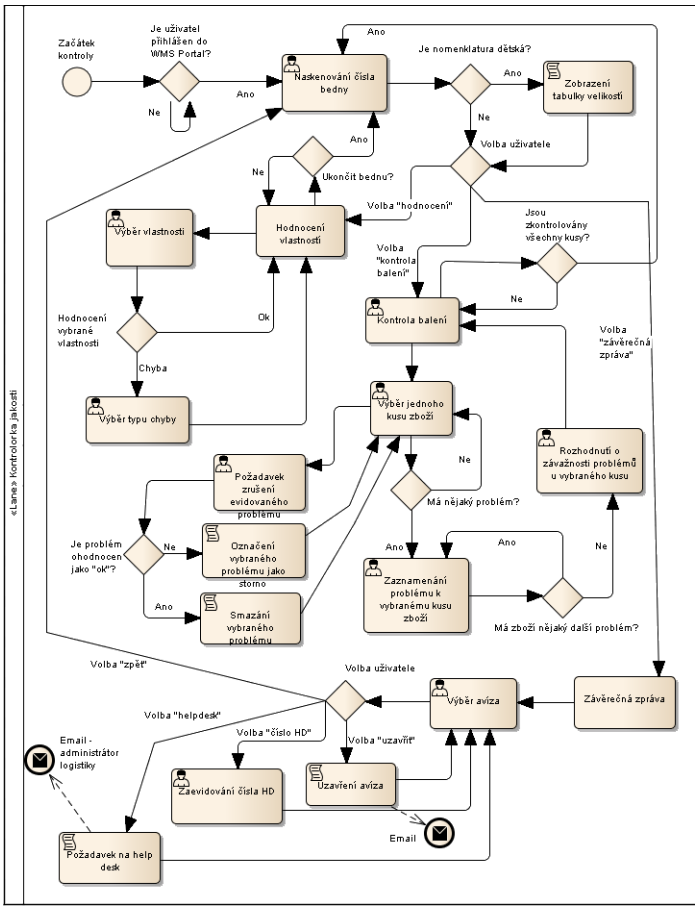
[15] Nick Russell, A.H.M. ter Hofstede, David Edmong *Workflow Resource Patterns*. Centre for Information Technology Innovation, Queensland University of Technology.

A Testy detekce vzorů v rozsáhlých modelech

Všechny testy jsou prováděny na počítačové sestavě s procesorem Intel i3-2120 3.3 Ghz, s pamětí 4 Gb RAM a se 64-bitovým systémem Windows 7. Čas výpočtu je celkovým souhrnem času potřebného k vyhledání všech obsažených vzorů v modelu.

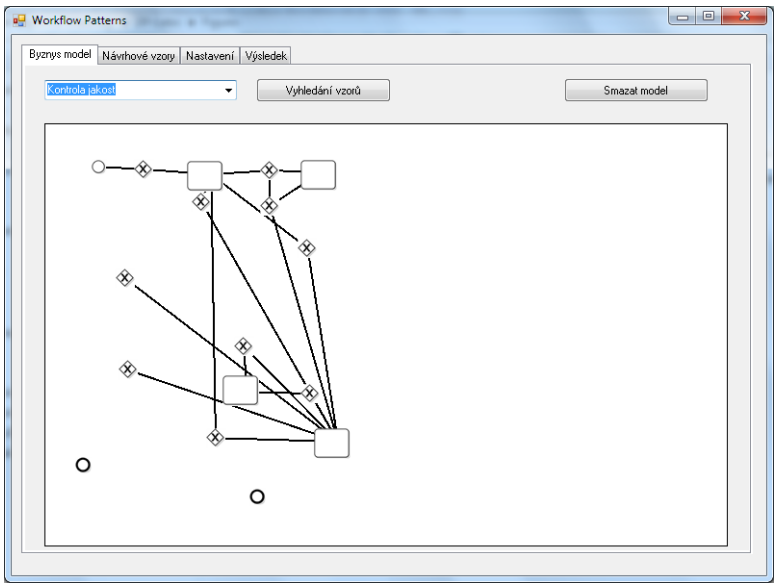
A.1 První ověření modelu z praxe

Pro vyzkoušení aplikace na ”skutečném”byznys modelu jsem použil model, který jsem vytvořil v rámci analýzy pro pracoviště kontroly jakosti textilního zboží, ve skladě logistiky. Průběh tohoto procesu začíná naskenováním čtečkou balení se zbožím, které pak kontrolorka rozbalí a postupně prochází jednotlivé kusy zboží, zda neobsahují nějakou vadu. Jestliže nějaký problém nalezne, zaeviduje požadavek k vyřešení na help desk společnosti a problém pak řeší administrátor ve spolupráci s obchodním oddělením. Model jsem vytvořil v aplikace Enterprise Architekt, bez zřetelu na procesní vzory.



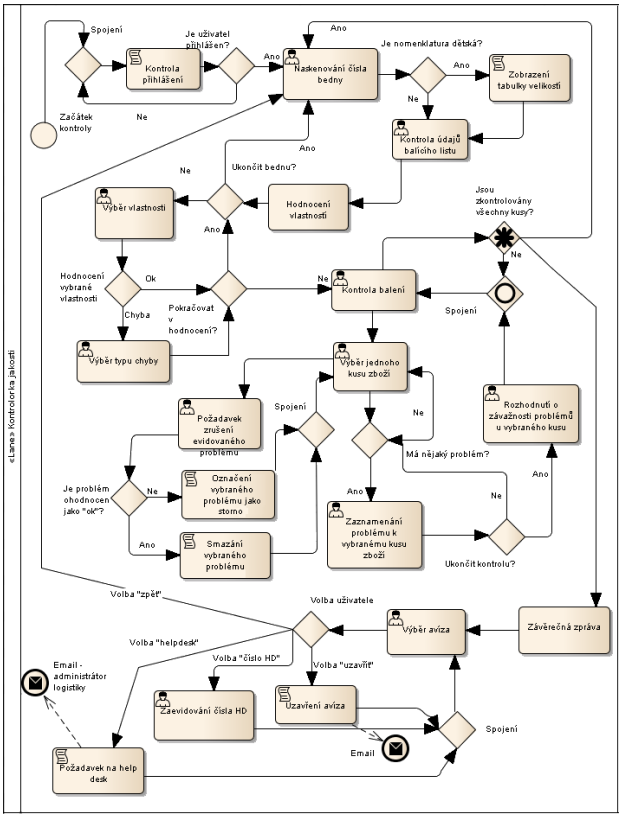
Obrázek 13: Byznys model kontroly kvality

Model jsem nechal zkontrolovat aplikací s použitím sady vzorů *Sequence*, *Structured Synchronizing Merge*, *Structured Loop* a *Multi Merge*. Po nalezení všech vzorů zůstaly na obrazovce aplikace prvky, jak je vidět na následujícím obrázku.

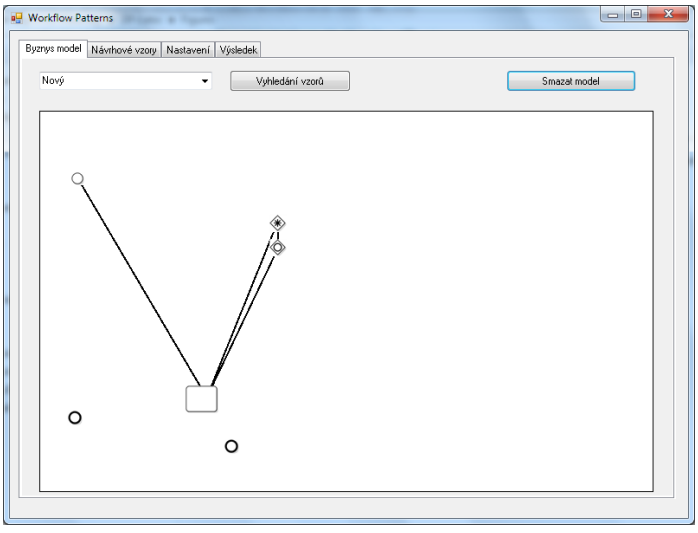


Obrázek 14: Výsledek ověření byznys modelu kontroly kvality v testovací aplikaci

Překvapilo mě velké množství bran, které zůstaly nevyřešeny. Když jsem se nad modelem hlouběji zamyslel, uvědomil jsem si, že vlastně v celém modelu nijak neřeším spojení procesů po předchozím rozdělení. Ačkoliv se většinou při rozdělení toku jedná o typy vzoru *Exclusive Choice*, který dovolí výběr jen jedné cesty, problém může nastat při paralelním zpracování dat v aktivitě, která přijme oba procesy současně (kontrolorka si vezme k ruce pomocnici, s vlastním počítačem). Také přihlášení uživatele po stránce formálního vyjádření není správně. Na základě těchto poznatků jsem model překreslil, tentokrát s využitím procesních vzorů a znovu ověřil pomocí testovací aplikace.



Obrázek 15: Upravený byznys model kontroly kvality

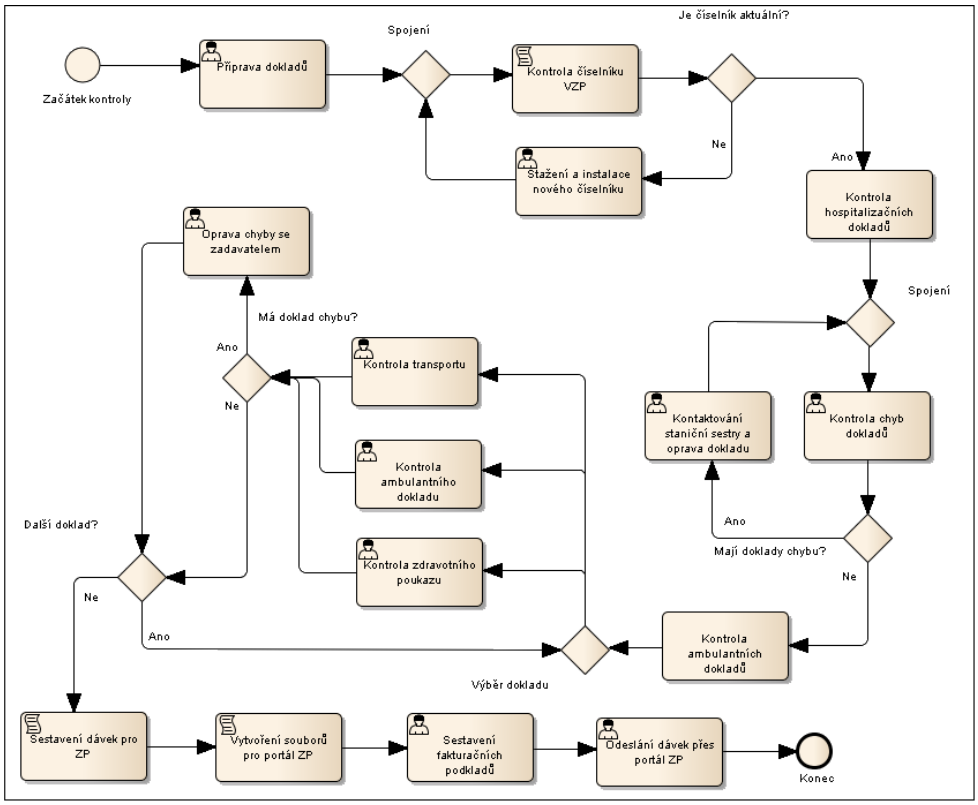


Obrázek 16: Nové ověření byznys modelu kontroly kvality v testovací aplikaci

Po odstranění všech vzorů zůstaly na obrazovce jen prvky, pro které už procesní vzor neexistuje. Celkový čas pro nalezení všech vzorů (8 vzorů *Sequence* a 7 vzorů *Multi Merge*) byl 3 minuty 50 sekund.

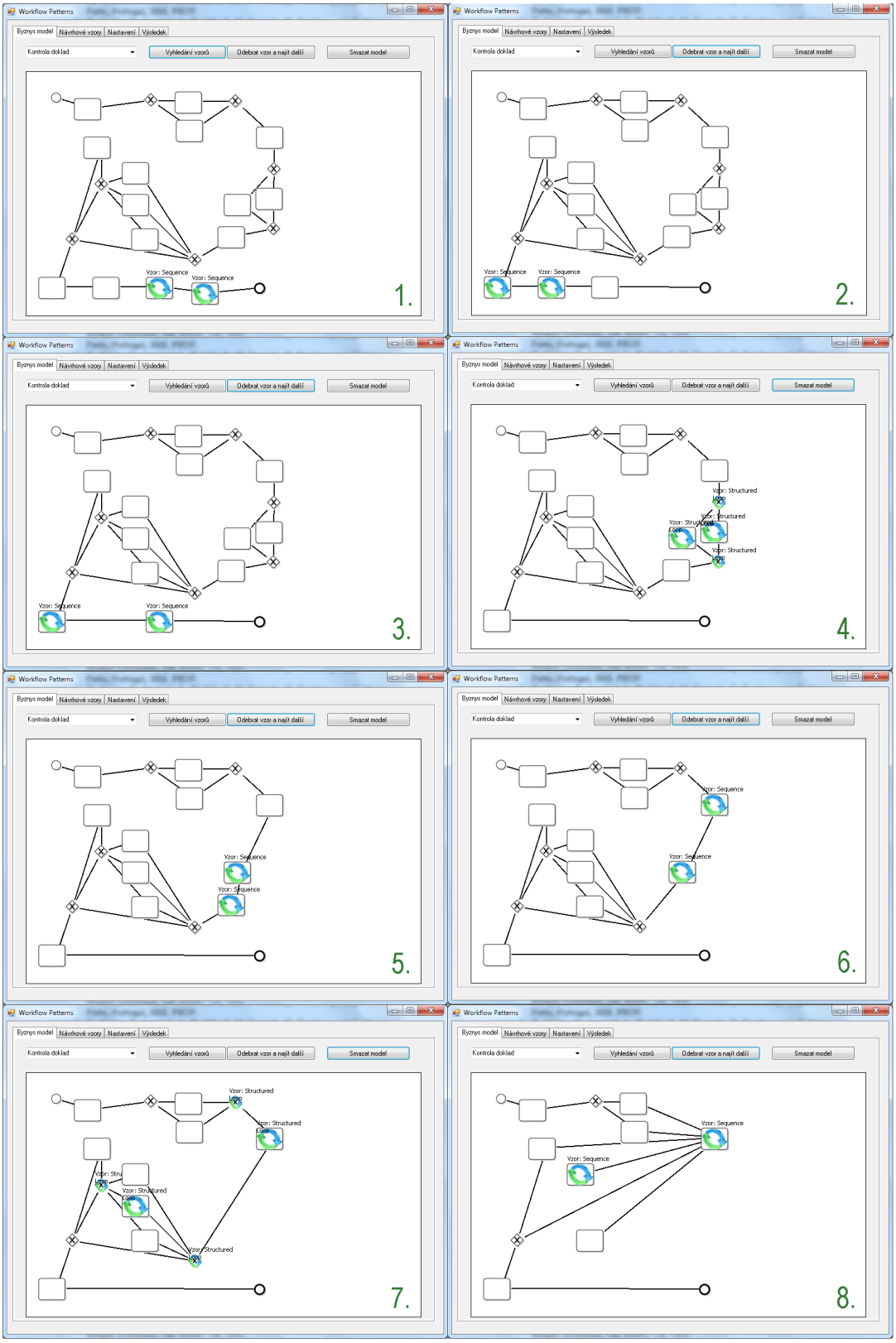
A.2 Test modelu kontroly dokladů

Pro další test jsem zvolil model procesu vytvoření dávek ambulantních a hospitalizačních dokladů rehabilitačního ústavu, které se následně odešlou pomocí webového rozhraní portálu zdravotním pojišťovnám. Z předběžné analýzy procesu vím, že se provádí několik kontrol, u kterých se regresně kontroluje každý doklad, ve kterém byla dříve nalezena nesrovnalost. Při modelování proto počítám s nasazením vzoru *Structured Loop*. Protože je dokladů několik druhů (hospitalizační, ambulantní, doprava, poukazy - rehabilitace) a každý z nich prochází stejnou kontrolou jako ostatní, ale samostatným procesem, jako další vzor k nasazení jsem zvolil *Exclusive Choice* a pro synchronizaci *Multi Merge*. Samozřejmostí je použití vzoru *Sequence*.

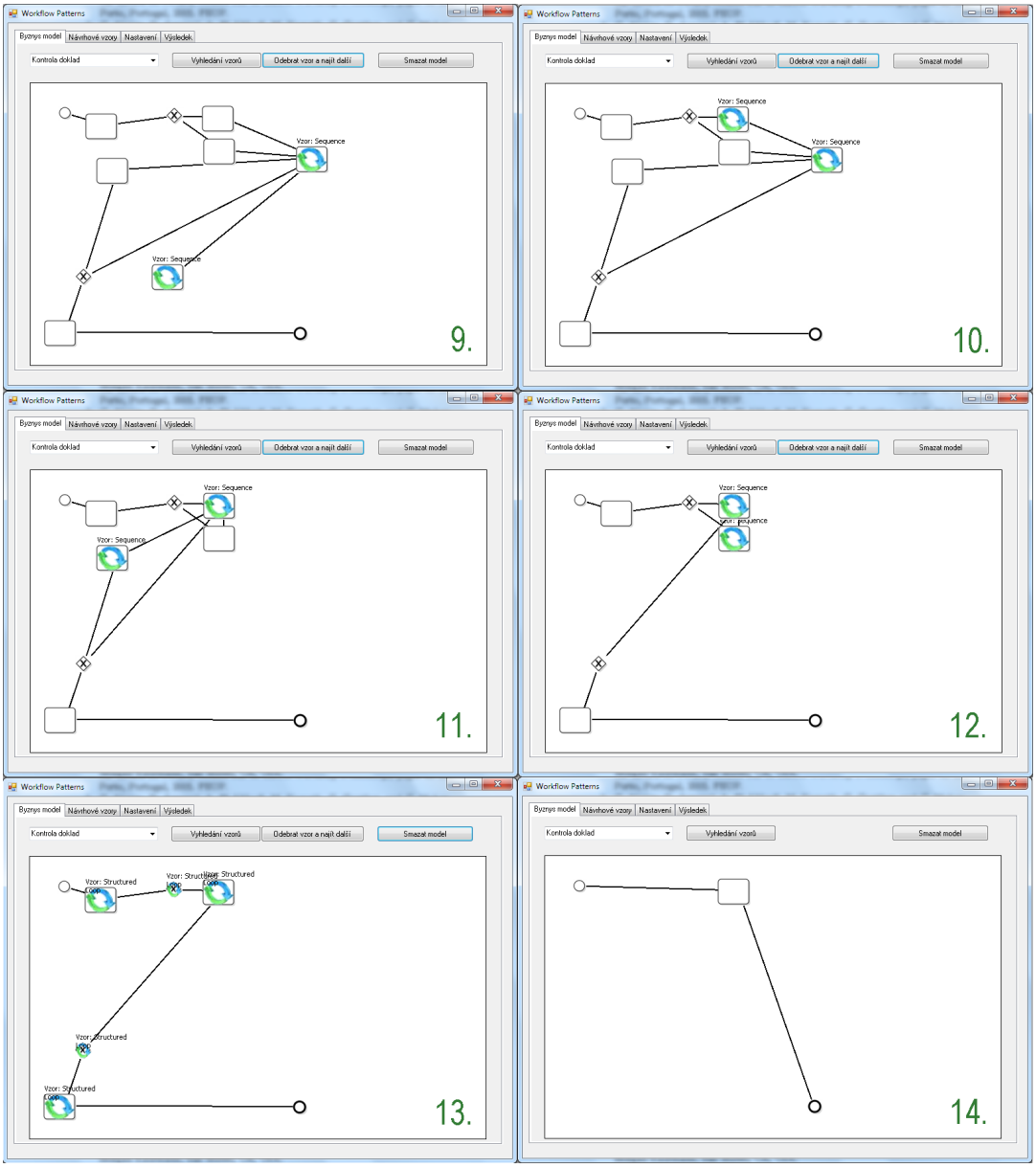


Obrázek 17: Byznys model kontroly dokladů

Průběh testu jsem zaznamenal po jednotlivých krocích, aby bylo názorně vidět, jak aplikace postupně odebírá jednotlivé vzory.



Obrázek 18: První část zobrazení kroků při hledání vzorů v modelu kontroly dokladů



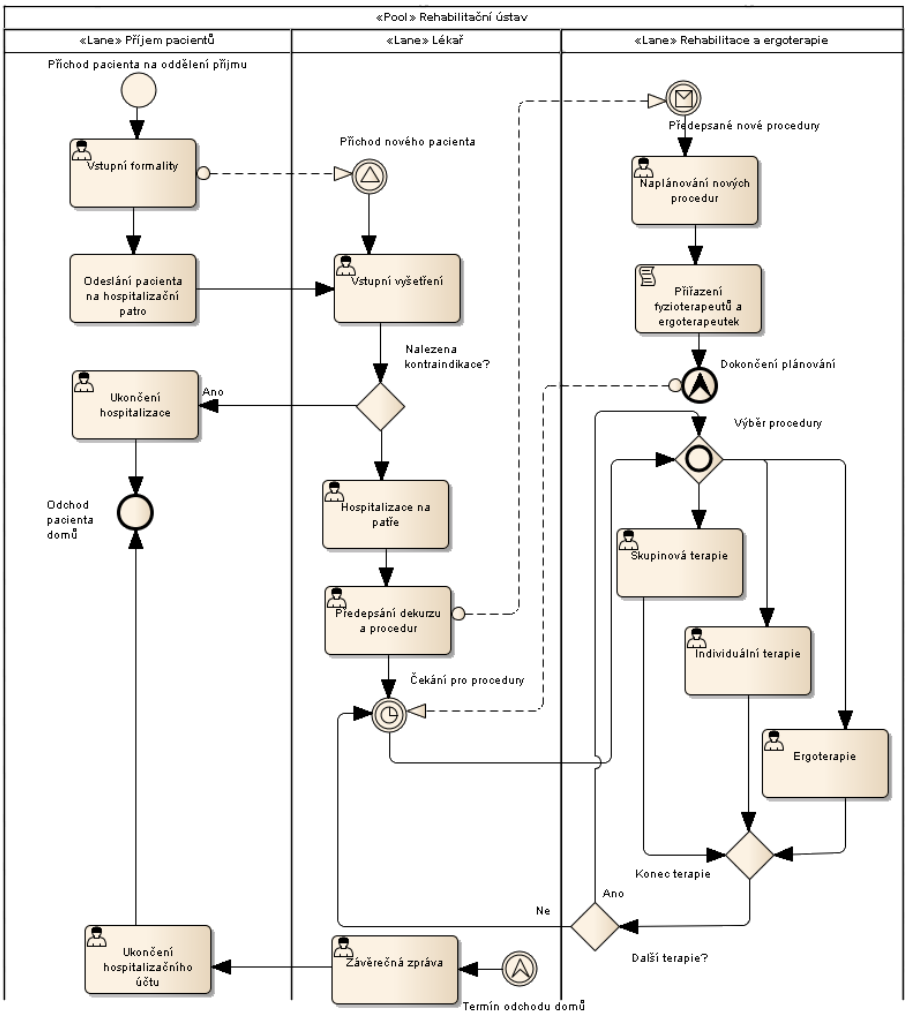
Obrázek 19: Pokračování zobrazení kroků při hledání vzorů v modelu kontroly dokladů

Jak je z průběhu testu vidět, aplikace úspěšně nalezla všechny vzory, které byly při modelování použity. Problém nastal jen u kroku č. 7 a pak 13, kde algoritmus nerozlišil správně rozložení prvků v rámci vzoru *Structured Loop*. Celkový čas pro nalezení všech vzorů byl 1 minuta, 10 sekund.

A.3 Test modelu hospitalizace pacienta

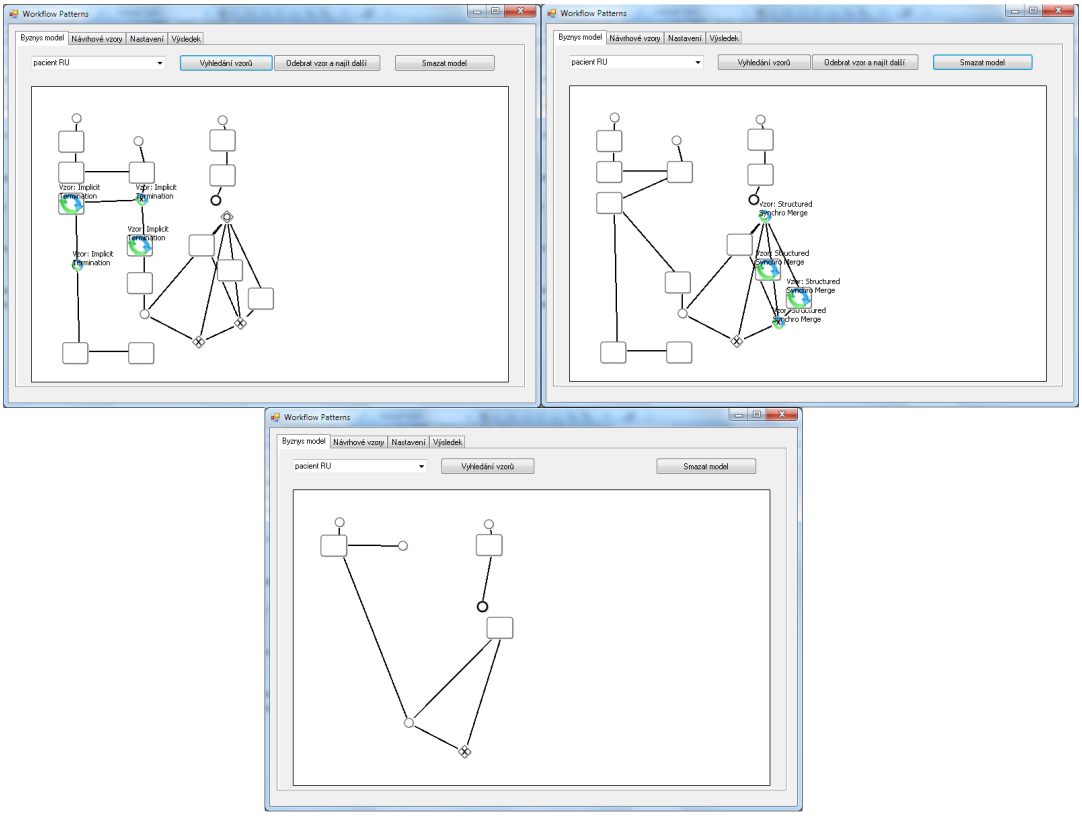
Ve svém třetím testu jsem se rozhodnul si to trochu zkomplikovat. Předmětem zkoumání je proces pobytu pacienta v rehabilitačním ústavu, kde postupně prochází oddělením příjmu, hospitalizačním patrem a oddělením rehabilitace. V modelu jsem použil prvky, které aplikace nezná a také členění na plavecké dráhy pro jednotlivá pracoviště. A byl jsem zvědavý, jak se s tím aplikace vypořádá.

Z procesních vzorů jsem si vybral *Implicit Termination* pro část procesu vstupního vyšetření lékařem, protože v této části může dojít k úplnému přerušení procesu - pokud lékař u pacienta nalezne kontraindikaci, pobyt automaticky bez možnosti odvolání končí. Dále jsem zvolil *Structured Synchronizing Merge* pro část na pracovišti rehabilitace, kde pacient postupně prochází několika naordinovanými procedurami.



Obrázek 20: Byznys model průběhu hospitalizace pacienta

Již při pokusu o import modelu jsem zjistil, že EA nedokáže najít prvky, které jsou podčástí jiných prvků, v tomto případě plaveckých drah. Poté, co jsem všechny prvky přesunul do zdrojového adresáře se import podařil a mohl být proveden test. Z něj jsem zaznamenal jen nalezení nejzajímavějších vzorů, vzory *Sequence* nejsou do přehledu zařazeny.



Obrázek 21: Vybrané části z průběhu hledání vzorů v modelu

Jak je vidět ve výběru přehledu, asociace *Message Flow* nejsou v aplikaci vůbec zaznamenány. Rozborem XML souboru modelu jsem zjistil, že EA pozice těchto propojení vůbec neexportuje. Nicméně, všechny vzory, použité při modelování procesu byly aplikací nalezeny. Čas, potřebný pro nalezení všech vzorů (1 vzor *Implicit Termination*, 1 vzor *Structured Synchronizing Merge* a 8 vzorů *Sequence*) je 2 minuty 18 sekund.

B Uživatelská příručka k aplikaci

Ke spuštění aplikace je potřeba splnit několik požadavků:

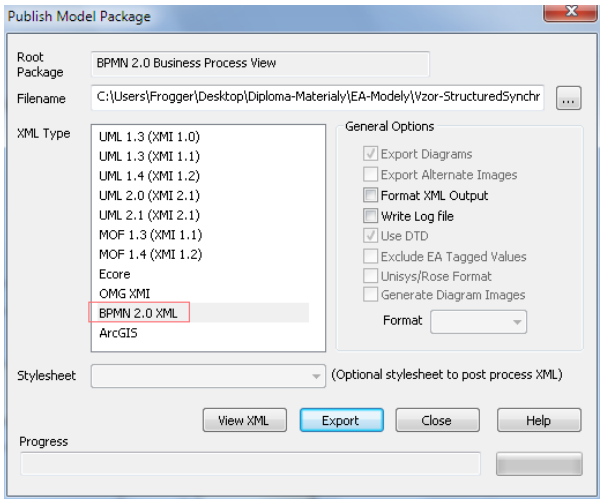
- Na počítači musí být nainstalovaný .NET Framework verze 4.0
- Ve stejném adresáři, jako je aplikace se nachází soubor pro uživatelskou konfiguraci $WorkflowPatterns.exe.config$ a soubor knihovny pro databázi SQLite.
- V hlavním adresáři disku D: se nachází soubor $wpdata.db$. Ten lze v případě potřeby přemístit na jiné místo na disku, jestliže se nové umístění zapíše do konfiguračního souboru aplikace (postup je vysvětlen v další kapitole příručky).

Aplikace se nemusí do systému instalovat, spouští se přímo. Po jejím spuštění jsou k dispozici 4 obrazovky, mezi kterými lze volně přepínat. Jestliže byly při posledním spuštění aplikace importovány vzory nebo modely a tyto uživatel uložil na disk (soubor $WPData.xml$), při spuštění aplikace se všechny automaticky do aplikace nahrají. Následuje popis jednotlivých obrazovek:

- **Nastavení** - tlačítkem "Načíst nový návrhový vzor/model z EA"je možné vybrat xml soubor, exportovaný z aplikace Enterprise Architect a ten je pak uložen do příslušné části aplikace. Všechny aktivní vzory a modely v aplikaci lze uložit na disk pomocí tlačítka "Uložit data na disk". Snadnou cestou, jak vymazat v případě potřeby všechny vzory a modely tak, aby se již nenačítaly je prosté vymazání souboru $WPData.xml$ z disku.
- **Procesní vzory** - rozbalovací menu obsahuje přehled všech procesních vzorů v aplikaci. Při importu nového vzoru dostane implicitní jméno "Nový"s případným vygenerovaným číslem pro zabránění duplicity názvu. Pro přejmenování vzoru stačí přepsat jeho název v rozbalovací nabídce a potvrdit volbu klávesou Enter. Tlačítkem "Smazat vzor"se odstraní vybraný vzor z aplikace (na disku ale zůstane uložen).
- **Byznys model** - obsahuje stejnou funkčnost pro práci s importovaným souborem, jako volba "Procesní vzory". Navíc je k dispozici tlačítko "Vyhledání vzorů", po jehož aktivaci začne aplikace porovnávat jednotlivé vzory s vybraným modelem. Při každém úspěšném nalezení vzoru se zpřístupní tlačítko "Odebrat vzor a najít další", které zajistí vyjmutí označených prvků v modelu a zahájí další hledání vzorů.
- **Výsledek** - při každém zahájení hledání vzorů v modelu, úspěšném nalezení vzoru a ukončení hledání se v této části zapíše informace o proběhlé události. Tlačítkem "Vyčistit"se záznam smaže.

B.1 Export dat z aplikace Enterprise Architect a uživatelská konfigurace

Model nebo vzor, který má být použit v aplikaci pro detekci vzorů musí být modelován jako projekt "BPMN 2.0 Business Process View". Pro samotný export je k dispozici funkce v menu "Projekt", volba "Model Publisher ...", kde je nutné z nabídky možností XML Type vybrat "BPMN 2.0 XML", tak jak je uvedeno na následujícím obrázku.



Obrázek 22: Volba exportu dat ve formátu XML v aplikaci Enterprise Architect

V souboru $WorkflowPatterns.exe.config$, který se nachází ve stejném adresáři, jako samotná aplikace je možné nastavit některé parametry pro běh programu:

- **appSettings** - obsahuje převážně definice prvků v importu XML souboru z EA.
  - **testPrvky** - rozlišení typů prvků v XML.
  - **flowPrvky** - rozlišení typu hrany v XML.
  - **facePrvky** - rozlišení části, ve které se nachází k prvku informace o velikosti a umístění v modelu (přiřazuje se podle značky ID).
  - **doplneniPrvky** - doplněk ke značce "bpmndi:BPMNShape"(někdy se zaměňují).
  - **dataPrvky** - rozlišení prvků typu "Data Object". Bohužel k těmto prvkům EA nepřikládá spojovací asociace s ostatními prvky, takže je nelze použít.
  - **sqlcesta** - odkaz na umístění databáze SQLite na disku.
- **GrafikaSettings** - přiřazení grafických prvků v aplikaci k jednotlivým typům, definovaných v klíči.
- **PrvkySettings** - výčet všech dostupných druhů prvků typu "activity", které se zamění za typ abstraktní úlohy.